MANUAL TECNICO

SGMDT de Wifimex

Miembros del equipo de desarrollo:

Juan Roman Pantoja Garcia

Alejandro Jiménez Ramírez

Rafael Carrillo Alcantar

ITSUR

IGN. Sistemas Computacionales

Contenido

INTRODUCION

En el presente manual tecnnico se mostrara el codigo fuente documentado, con el fin de hacer mas facil la lectura y en caso posteriores tenerner una foma mas facil de poder trabajar con el codigo.

Con el fin de hacer mas facil el trabajar con el codigo fuente y hacer mas facil como funciona en cada modulo con sus respectivos submodulos tiene una breve explicacion de que es lo que hace cada uno y a si mismo como funciona cada uno de los parametros dentro de cada submodulo.

CLASE DE CONEXIÓN.

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MySql.Data.MySqlClient;

namespace Datos
{
 /// <summary>
 /// Clase En la Cual se definen lass variables y un metodo que permitira la conexion hacia la base de datos
///  del programa.
/// </summary>
     public class Conexion
     {
        /// <summary>
        /// MySqlConnection variable que permite la conexión hacia la base de datos.
        /// usuario variable tipo string que almacena el nombre de usuario de la base de datos.
        /// password variable tipo string que almacena la contraseña de la base de datos.
        /// bd variable de tipo string que almacena el nombre de la base de datos.
        /// servidor variable de tipo string que almacena el nombre del servidor de la base de datos.
        /// puerto variable de tipo string que almacena el puerto en el que esta el servidor que almacena
        /// la base de datos.
        /// </summary>
           public static MySqlConnection conexion;
           public static MySqlTransaction transaccion;
           static string usuario = "root";
           static string password = "12345";
           static string bd = "dbwifimex";
           static string servidor = "localhost";
           static string puerto = "3306";


            /// <summary>
        /// Método de Conexión en este método se define la conexión hacia la base de datos que contiene la información
        /// sobre la empresa a la que se le realizo el programa
        /// </summary>
        /// <returns>
        /// Regresa un valor Booleano de verdadero en caso de que la conexión se pudiera realizar con éxito.
        /// Regresa un valor Booleano de falso en caso de que la conexión  no se pudo establecer.
        /// </returns>
            public static bool conectar()
         {
             try
             {
                 conexion = new MySqlConnection("Database=" + bd + ";Data Source=" + servidor
                 + ";Port=" + puerto + ";User Id=" + usuario + ";Password=" + password);

                  conexion.Open();
                  return true;
             }
             catch (Exception)
             {
                 return false;
             }
         }
         public static void desconectar()
         {
             try
             {
                 conexion.Close();
             }
             catch (Exception)
             {

             }
         }
     }
}
```

DAOAlmancen

```
using Modelos;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Datos
{
    /// <summary>
    /// Clase DAOAlamacen que contiene los métodos para realizar consultas, inserciones, actualizaciones y eliminación
    /// de registros de la tabla Almacén de la base de datos del programa.
    /// </summary>
    public class DAOAlmacen
    {
        /// <summary>
        /// Metodo para obtener una list con todos los registros que estén almacenados en la base de datos
        /// del programa en la tabla de almacén.
        /// </summary>
        /// <returns>
        /// Este método realiza una conexion hacia la base de datos para obtener una lista con todos
        /// los registros que tenga la tabla de almacén y regresa una lista de tipo Almacén.
        /// </returns>
        /// <exception cref="Exception">
        /// Encaso de no poder encontrar o establecer una conexión hacia la base de datos
        /// mandara un mensaje de que no se pudo realizar la conexión hacia la base de datos.
        /// En caso de que no se pudo encontrar el registros a buscar regresara un mensaje
        /// de que no se pudo encontrar el registro.
        /// </exception>
public List<Almacen> ObtenerTodos()
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand(
                        @"SELECT * FROM almacen");
                    comando.Connection = Conexion.conexion;
                    MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                    DataTable resultado = new DataTable();
                    adapter.Fill(resultado);
                    List<Almacen> lista = new List<Almacen>();
                    Almacen objAlmacen = null;

                    foreach (DataRow fila in resultado.Rows)
                    {
                        objAlmacen = new Almacen();
                        objAlmacen.idAlmacen = fila["idAlmacen"].ToString();
                        objAlmacen.cantProducto = int.Parse(fila["cantProducto"].ToString());
                        objAlmacen.codigoBarra = fila["codigoBarra"].ToString();
                        objAlmacen.idEmpleado = fila["idEmpleado"].ToString();
                        objAlmacen.Estatus = fila["Estatus"].ToString() == "True" ? "Activo" : "Inactivo";

                        lista.Add(objAlmacen);
                    }

                    return lista;
                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                throw new Exception("No se pudo obtener la información de los almacenes");
            }
            finally
            {
                Conexion.desconectar();
            }
        }
         /// <summary>
        /// Método que recibe un parámetro de entrada para la búsqueda de un registro en específico
        /// en la de Almacén para posteriormente regresar el resultado.
        /// </summary>
        /// <param name="text">
        /// Variable de tipo string que se recibe como entrada en el método la cual almacena
        /// un texto para la búsqueda de un registro en la base de datos del programa.
        /// </param>
        /// <returns>
```

```
/// Regresa una Lista de tipo Almacén con el registro que coincida con el texto de la variable
/// si se encuentra en la base de datos.
/// </returns>
/// <exception cref="Exception">
/// Encaso de no poder encontrar o establecer una conexión hacia la base de datos
/// mandara un mensaje de que no se pudo realizar la conexión hacia la base de datos.
/// En caso de que no se pudo encontrar el registros a buscar regresara un mensaje
/// de que no se pudo encontrar el registro.
/// </exception>
public List<Almacen> Buscar(string text)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand(
                @"select * from almacen
                      where
                      codigoBarra like  CONCAT('%', @ref, '%') ||
                      idEmpleado like  CONCAT('%', @ref, '%') ||
                      idAlmacen like  CONCAT('%', @ref, '%') ||
                      cantProducto like  CONCAT('%', @ref, '%');"
            );
            comando.Parameters.AddWithValue("@ref", text);
            comando.Connection = Conexion.conexion;
            MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
            DataTable resultado = new DataTable();
            adapter.Fill(resultado);
            List<Almacen> lista = new List<Almacen>();
            Almacen objAlmacen = null;

            foreach (DataRow fila in resultado.Rows)
            {
                objAlmacen = new Almacen();
                objAlmacen.idAlmacen = fila["idAlmacen"].ToString();
                objAlmacen.cantProducto = int.Parse(fila["cantProducto"].ToString());
                objAlmacen.codigoBarra = fila["codigoBarra"].ToString();
                objAlmacen.idEmpleado = fila["idEmpleado"].ToString();
                objAlmacen.Estatus = fila["Estatus"].ToString() == "True" ? "Activo" : "Inactivo";

                lista.Add(objAlmacen);
            }

            return lista;
        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        throw new Exception("No se pudo obtener la información de los almacenes");
    }
    finally
    {
        Conexion.desconectar();
    }
}
/// <summary>
/// Metodo que permite obtener un registro de la base de datos del programa de la tabla almacén
/// a través del ID de un almacén proporcionado por el usuario.
/// </summary>
/// <param name="id">
/// id variable de tipo String que almacena el ID de un almacén para realizar una búsqueda en la base de datos
/// a través de ID
/// </param>
/// <returns>
/// Regresa una variable de tipo Almacén que tendría almacenado los datos de un registro
/// de la tabla Almacén que coincidan con el id proporcionado.
/// </returns>
/// <exception cref="Exception">
/// Encaso de no poder encontrar o establecer una conexión hacia la base de datos
/// mandara un mensaje de que no se pudo realizar la conexión hacia la base de datos.
/// En caso de que no se pudo encontrar el registros a buscar regresara un mensaje
/// de que no se pudo encontrar el registro.
/// </exception>
public Almacen ObtenerUno(string id)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand(@"SELECT * FROM almacen WHERE idAlmacen=@id");
            comando.Parameters.AddWithValue("@id", id);
            comando.Connection = Conexion.conexion;
            MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
            DataTable resultado = new DataTable();
            adapter.Fill(resultado);

            Almacen objAlmacen = null;
            if (resultado.Rows.Count > 0)
            {
```

```
                DataRow fila = resultado.Rows[0];
                objAlmacen = new Almacen();
                objAlmacen.idAlmacen = fila["idAlmacen"].ToString();
                objAlmacen.cantProducto = int.Parse(fila["cantProducto"].ToString());
                objAlmacen.codigoBarra = fila["codigoBarra"].ToString();
                objAlmacen.idEmpleado = fila["idEmpleado"].ToString();
                objAlmacen.Estatus = fila["Estatus"].ToString() == "True" ? "Activo" : "Inactivo";
            }

            return objAlmacen;
        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        throw new Exception("No se pudo obtener la información del almacen");
    }
    finally
    {
        Conexion.desconectar();
    }
}
/// <summary>
/// Método que permite Eliminar un registro de la base de datos de la tabla Almacén
/// mediante un variable ID que permite la búsqueda de registro
/// </summary>
/// <param name="id">
/// id vairable de tipo String que almacena el ID de un registro de la tabla Almacén
/// que permitirá la búsqueda de un registro en la base de datos para su posterior eliminación.
/// </param>
/// <returns>
/// Regresa una variable de tipo Booleano si esta es variable tiene el valor de 1 o True
/// indicara que el registro se pudo eliminar, si la variable tiene el valor de 0 o false
/// indicara que el registro no pudo eliminarse.
/// </returns>
/// <exception cref="Exception">
/// Encaso de no poder encontrar o establecer una conexión hacia la base de datos
/// mandara un mensaje de que no se pudo realizar la conexión hacia la base de datos.
/// En caso de que no se pudo encontrar el registros a buscar regresara un mensaje
/// de que no se pudo encontrar el registro.
/// </exception>
public bool eliminar(string id)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand(
                @"DELETE FROM almacen
                WHERE idAlmacen=@id");

            comando.Parameters.AddWithValue("@id", id);
            comando.Connection = Conexion.conexion;
            int filasBorradas = comando.ExecuteNonQuery();
            return (filasBorradas > 0);
        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        if (ex.Number == 1451)
        {
            MySqlCommand comando = new MySqlCommand(
                @"update almacen set Estatus=0
                WHERE idAlmacen=@id");

            comando.Parameters.AddWithValue("@id", id);
            comando.Connection = Conexion.conexion;
            int filasBorradas = comando.ExecuteNonQuery();
            return (filasBorradas > 0);
        }
        else
        {
            throw new Exception("Error al intentar eliminar el Almacen");
        }

    }
    finally
    {
        Conexion.desconectar();
    }
}
/// <summary>
/// Método que permite Actualizar un registro de la tabla Almacén.
/// </summary>
/// <param name="almacen">
/// Parámetro de entrada de tipo Almacén que contiene todos los datos de un registro
```

```
/// de la base de datos de la tabla Almacén.
/// </param>
/// <returns>
/// Regresa una variable de tipo Booleano que contendrá un valor de 1 o True cuando
/// la actualizacion del registro se realizó con éxito, y tendrá un valor de 0 o false
/// en caso de que no se pudo realizar.
/// </returns>
/// <exception cref="Exception">
/// En caso de no poder encontrar o establecer una conexión hacia la base de datos
/// mandara un mensaje de que no se pudo realizar la conexión hacia la base de datos.
/// En caso de que no se pudo Editar o actualizar el registro regresa un mensaje
/// donde indica que la actualización no se logró realizar.
/// </exception>
public bool editar(Almacen almacen)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand(
                @"UPDATE `dbwifimex`.`almacen` SET
                `cantProducto` = @cant,
                `codigoBarra` = @codi,
                `idEmpleado` = @emp,
                `Estatus` = @estatus
                WHERE (`idAlmacen` = @id);");

            comando.Parameters.AddWithValue("@codi", almacen.codigoBarra);
            comando.Parameters.AddWithValue("@cant", almacen.cantProducto);
            comando.Parameters.AddWithValue("@emp", almacen.idEmpleado);
            comando.Parameters.AddWithValue("@id", almacen.idAlmacen);
            comando.Parameters.AddWithValue("@estatus", almacen.Estatus == "Activo" ? 1 : 0);

            comando.Connection = Conexion.conexion;

            int filasBorradas = comando.ExecuteNonQuery();

            return (filasBorradas > 0);
        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        throw new Exception("No se pudo editar la info del almacen");
    }
    finally
    {
        Conexion.desconectar();
    }
}
/// <summary>
/// Método que permite Agregar un registro en la base de datos en la tabla de almacén.
/// </summary>
/// <param name="almacen">
/// Parámetro de tipo Almacén que recibe todos los datos que se almacenaran
/// en el nuevo registro de la tabla almacén de la base de datos del programa
/// </param>
/// <returns>
/// Regresa una variable de tipo int con el valor de 1 si el registro se realizó con éxito
/// y con un valor de tipo 0 cuando no registro no se pudo realizar.
/// </returns>
/// <exception cref="Exception">
/// En caso de no poder encontrar o establecer una conexión hacia la base de datos
/// mandara un mensaje de que no se pudo realizar la conexión hacia la base de datos.
/// En caso de que no se pudo Agregar o insertar el registro regresa un mensaje
/// donde indica que el registro no se logró realizar.
/// </exception>
public int agregar(Almacen almacen)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand(
                "INSERT INTO `dbwifimex`.`almacen` " +
                "(`idAlmacen`, `cantProducto`, `codigoBarra`, `idEmpleado`, `Estatus`) VALUES " +
                "(@id, @cant, @codi, @emp, '1')");

            comando.Parameters.AddWithValue("@id", almacen.idAlmacen);
            comando.Parameters.AddWithValue("@cant", almacen.cantProducto);
            comando.Parameters.AddWithValue("@codi", almacen.codigoBarra);
            comando.Parameters.AddWithValue("@emp", almacen.idEmpleado);
            comando.Connection = Conexion.conexion;

            int filasAgregadas = comando.ExecuteNonQuery();

            return filasAgregadas;
        }
        else
        {
```

```
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                throw new Exception("No se pudo agregar, intentelo de nuevo o comuniquese con el administrador");
            }
            finally
            {
                Conexion.desconectar();
            }
        }
    }
}
```

DAOCliente

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Modelos;
using MySql.Data.MySqlClient;

namespace Datos
{
    /// <summary>
    /// Clase DAOCliente que contiene los métodos para realizar consultas, inserciones, actualizaciones y eliminación
    /// de registros de la tabla Clientes de la base de datos del programa.
    /// </summary>
    public class DAOCliente
    {
        /// <summary>
        /// Método que Realiza una consulta en la tabla de Clientes para obtener todos los
        /// registros de la tabla.
        /// </summary>
        /// <returns>
        /// Regresa un Lista de tipo Cliente que contiene todos los registros de la tabla de clientes.
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de que no se pudo realizar la conexión se mandara un mensaje donde indica que la
        /// conexión no se logró realizar.
        /// En caso de que no se encuentren datos en la tabla de clientes mandara un mensaje que
        /// indica que no se pudieron obtener dato de la tabla.
        /// </exception>
        public List<Cliente> ObtenerTodos()
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand(
                        @"SELECT * FROM clientes");
                    comando.Connection = Conexion.conexion;
                    MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                    DataTable resultado = new DataTable();
                    adapter.Fill(resultado);
                    List<Cliente> lista = new List<Cliente>();
                    Cliente objCliente = null;

                    foreach (DataRow fila in resultado.Rows)
                    {
                        objCliente = new Cliente();
                        objCliente.idCliente = fila["idCliente"].ToString();
                        objCliente.RFC = fila["RFC"].ToString();
                        objCliente.CURP = fila["CURP"].ToString();
                        objCliente.nomCliente = fila["nomCliente"].ToString();
                        objCliente.Direccion = fila["Direccion"].ToString();
                        objCliente.Telefono = fila["Telefono"].ToString();
                        objCliente.Correo = fila["Correo"].ToString();
                        objCliente.Estatus = fila["Estatus"].ToString()=="True"?"Activo":"Inactivo";

                        lista.Add(objCliente);
                    }

                    return lista;
                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                throw new Exception("No se pudo obtener la información de los clientes");
            }
            finally
            {
                Conexion.desconectar();
            }
        }
        /// <summary>
        /// Metodo que Realiza consultas para obtener registros de la tabla clientes.
        /// </summary>
        /// <param name="text">
        /// Parámetro de entrada que recibe una variable de tipo String que contiene una cadena
        /// de texto que se utilizara para buscar coincidencias con los registros de la tabla clientes.
        /// </param>
        /// <returns>
        /// Regresa una lista de registros de la tabla clientes que coincidan con la cadena de texto que
        /// se proporciono.
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de que no se pudo realizar la conexión se mandara un mensaje donde indica que la
        /// conexión no se logro realizar.
```

```
/// En caso de que no se encuentren datos en la tabla de clientes mandara un mensaje que
/// indica que no se pudieron obtener dato de la tabla.
/// </exception>
public List<Cliente> Buscar( string text)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand(
                @"select * from clientes
                      where
                      idCliente like  CONCAT('%', @ref, '%') ||
                      RFC like  CONCAT('%', @ref, '%') ||
                      CURP like  CONCAT('%', @ref, '%') ||
                      nomCliente like  CONCAT('%', @ref, '%') ||
                      Direccion like  CONCAT('%', @ref, '%') ||
                      Telefono like  CONCAT('%', @ref, '%') ||
                      Correo like  CONCAT('%', @ref, '%');"
            );
            comando.Parameters.AddWithValue("@ref", text);
            comando.Connection = Conexion.conexion;
            MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
            DataTable resultado = new DataTable();
            adapter.Fill(resultado);
            List<Cliente> lista = new List<Cliente>();
            Cliente objCliente = null;

            foreach (DataRow fila in resultado.Rows)
            {
                objCliente = new Cliente();
                objCliente.idCliente = fila["idCliente"].ToString();
                objCliente.RFC = fila["RFC"].ToString();
                objCliente.CURP = fila["CURP"].ToString();
                objCliente.nomCliente = fila["nomCliente"].ToString();
                objCliente.Direccion = fila["Direccion"].ToString();
                objCliente.Telefono = fila["Telefono"].ToString();
                objCliente.Correo = fila["Correo"].ToString();
                objCliente.Estatus = fila["Estatus"].ToString() == "True" ? "Activo" : "Inactivo";

                lista.Add(objCliente);
            }

            return lista;
        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        throw new Exception("No se pudo obtener la información de los clientes");
    }
    finally
    {
        Conexion.desconectar();
    }
}
/// <summary>
/// Método que realiza una consulta para obtener un registro en base a id proporcionado
/// </summary>
/// <param name="id">
/// Parámetro que recibe un variable de tipo String con la que se realizara una búsqueda
/// en la tabla de clientes
/// </param>
/// <returns>
/// regresa un objeto de tipo cliente que contendrá toda la información del registro
/// que coincida con el id proporcionado.
/// </returns>
/// <exception cref="Exception">
/// En caso de que no se pudo realizar la conexión se mandara un mensaje donde indica que la
/// conexión no se logró realizar.
/// En caso de que no se encuentren datos en la tabla de clientes mandara un mensaje que
/// indica que no se pudieron obtener dato de la tabla.
/// </exception>
public Cliente ObtenerUno(String id)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand(@"SELECT * FROM clientes WHERE idCliente=@id");
            comando.Parameters.AddWithValue("@id", id);
            comando.Connection = Conexion.conexion;
            MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
            DataTable resultado = new DataTable();
            adapter.Fill(resultado);

            Cliente objCliente = null;
            if (resultado.Rows.Count > 0)
            {
                DataRow fila = resultado.Rows[0];
```

```
                objCliente = new Cliente();
                objCliente.idCliente = fila["idCliente"].ToString();
                objCliente.RFC = fila["RFC"].ToString();
                objCliente.CURP = fila["CURP"].ToString();
                objCliente.nomCliente = fila["nomCliente"].ToString();
                objCliente.Direccion = fila["Direccion"].ToString();
                objCliente.Telefono = fila["Telefono"].ToString();
                objCliente.Correo = fila["Correo"].ToString();
                objCliente.Estatus = fila["Estatus"].ToString() == "True" ? "Activo" : "Inactivo";
            }

            return objCliente;
        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        throw new Exception("No se pudo obtener la información del cliente");
    }
    finally
    {
        Conexion.desconectar();
    }
}
/// <summary>
/// Método que permite eliminar un registro de la tabla cliente mediante un id proporcionado.
/// </summary>
/// <param name="id">
/// Parámetro que recibe una variable de tipo String que contiene un id para realizar un
/// consulta que coincida con el id proporcionado.
/// </param>
/// <returns>
/// Regresara una variable de tipo Boolean con el valor 1 o true si el registro se eliminó con éxito.
/// en caso de lo contrario regresara un valor 0 o false si no se logró eliminar el registro.
/// </returns>
/// <exception cref="Exception">
/// En caso de que no se pudo realizar la conexión se mandara un mensaje donde indica que la
/// conexión no se logró realizar.
/// En caso de que no se encuentre el registro o no se pueda eliminar mandara un mensaje donde indica
/// que no se pudo eliminar el registro.
/// </exception>
public bool eliminar(string id)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand(
                @"DELETE FROM clientes
                WHERE idCliente=@id");

            comando.Parameters.AddWithValue("@id", id);
            comando.Connection = Conexion.conexion;
            int filasBorradas = comando.ExecuteNonQuery();
            return (filasBorradas > 0);
        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        if (ex.Number == 1451)
        {
            MySqlCommand comando = new MySqlCommand(
                @"update clientes set Estatus=0
                WHERE idCliente=@id");

            comando.Parameters.AddWithValue("@id", id);
            comando.Connection = Conexion.conexion;
            int filasBorradas = comando.ExecuteNonQuery();
            return (filasBorradas > 0);
        }
        else
        {
            throw new Exception("Error al intentar eliminar al cliente");
        }

    }
    finally
    {
        Conexion.desconectar();
    }
}
/// <summary>
/// Método que permite Actualizar registros de la tabla clientes.
/// </summary>
/// <param name="cliente">
/// Parámetro de tipo cliente que recibirá todos los datos del registro seleccionado para su actualización.
```

```
/// </param>
/// <returns>
/// Regresa una variable de tipo Boolean con un valor de 1 o true cuando el registro se pudo realizar con éxito,
/// en caso de lo contrario regresara un valor 0 o false.
/// </returns>
/// <exception cref="Exception">
/// En caso de que no se pudo realizar la conexión se mandara un mensaje donde indica que la
/// conexión no se logró realizar.
/// En caso de que la actualización no se pudo realizar mandara un mensaje que indique que no se pudo
/// realizar la actualización del registro.
/// </exception>
public bool editar(Cliente cliente)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand(
                @"UPDATE `dbwifimex`.`clientes` SET
                `RFC` = @rfc,
                `CURP` = @curp,
                `nomCliente` = @name,
                `Direccion` = @dir,
                `Telefono` = @num,
                `Correo` = @corr,
                `Estatus` = @estatus
                WHERE (`idCliente` = @id);");

            comando.Parameters.AddWithValue("@id", cliente.idCliente);
            comando.Parameters.AddWithValue("@rfc", cliente.RFC);
            comando.Parameters.AddWithValue("@curp", cliente.CURP);
            comando.Parameters.AddWithValue("@name", cliente.nomCliente);
            comando.Parameters.AddWithValue("@dir", cliente.Direccion);
            comando.Parameters.AddWithValue("@num", cliente.Telefono);
            comando.Parameters.AddWithValue("@corr", cliente.Correo);
            comando.Parameters.AddWithValue("@estatus", cliente.Estatus=="Activo"?1:0);

            comando.Connection = Conexion.conexion;

            int filasBorradas = comando.ExecuteNonQuery();

            return (filasBorradas > 0);
        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        throw new Exception("No se pudo editar la info del cliente");
    }
    finally
    {
        Conexion.desconectar();
    }
}
/// <summary>
/// Metodo que permite agregar registros a la tabla de clientes.
/// </summary>
/// <param name="cliente">
/// Parámetro tipo cliente que recibe todos los datos del nuevo registro para su posterior
/// almacenamiento de registro.
/// </param>
/// <returns>
/// Regresa una variable de tipo int con el valor de 1 si el registro se realizó con éxito
/// y o en caso de que el registro no se logró realizar.
/// </returns>
/// <exception cref="Exception">
/// En caso de que no se pudo realizar la conexión se mandara un mensaje donde indica que la
/// conexión no se logró realizar.
/// En caso de que la inserción no se pudo realizar mandara un mensaje que indique que no se pudo
/// realizar la agregación del registro.
/// </exception>
public int agregar(Cliente cliente)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand(
                "INSERT INTO `dbwifimex`.`clientes` " +
                "(`idCliente`, `RFC`, `CURP`, `nomCliente`, `Direccion`, `Telefono`, `Correo`, `Estatus`) VALUES" +
                "(@id, @rfc, @curp, @name, @dir, @num, @corr, '1');");

            comando.Parameters.AddWithValue("@id", cliente.idCliente);
            comando.Parameters.AddWithValue("@rfc", cliente.RFC);
            comando.Parameters.AddWithValue("@curp", cliente.CURP);
            comando.Parameters.AddWithValue("@name", cliente.nomCliente);
            comando.Parameters.AddWithValue("@dir", cliente.Direccion);
            comando.Parameters.AddWithValue("@num", cliente.Telefono);
            comando.Parameters.AddWithValue("@corr", cliente.Correo);
            comando.Connection = Conexion.conexion;
```

```
                int filasAgregadas = comando.ExecuteNonQuery();

                return filasAgregadas;
            }
            else
            {
                return -1;
                throw new Exception("No se ha podido conectar con el servidor");
            }
        }
        catch (MySqlException ex)
        {
            throw new Exception("No se pudo agregar, intentelo de nuevo o comuniquese con el administrador");
        }
        finally
        {
            Conexion.desconectar();
        }
    }

}
}
```

DAOContratos

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Diagnostics.Contracts;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Modelos;
using MySql.Data.MySqlClient;

namespace Datos
{
    /// <summary>
    /// Clase DAOContratos que contiene los métodos para realizar consultas, inserciones, actualizaciones y eliminación
    /// de registros de la tabla Contratos de la base de datos del programa.
    /// </summary>
    public class DAOContratos
    {

        /// <summary>
        /// Método que Obtiene todos los registros de la tabla de Contratos.
        /// </summary>
        /// <returns>
        /// Regresa una lista con todos los registros de la tabla de Contratos
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de que no se puede conectar al servidor de la base de datos se mandara un mensaje
        /// que indica que no se pudo conectar al servidor.
        /// En caso de que no se pueda encontrar los registros mandara un mensaje indicando donde
        /// indica que no se encontraron registros.
        /// </exception>
        public List<Contrato> ObtenerTodos()
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand(
                        @"SELECT * FROM contratos");
                    comando.Connection = Conexion.conexion;
                    MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                    DataTable resultado = new DataTable();
                    adapter.Fill(resultado);
                    List<Contrato> lista = new List<Contrato>();
                    Contrato objContrato = null;

                    foreach (DataRow fila in resultado.Rows)
                    {
                        objContrato = new Contrato();
                        objContrato.idContrato = fila["idContrato"].ToString();
                        objContrato.precioServicio = double.Parse(fila["precioServicio"].ToString());
                        objContrato.inicioContrato = fila["inicioContrato"].ToString();
                        objContrato.finContrato = fila["finContrato"].ToString();
                        objContrato.idCliente = fila["idCliente"].ToString();
                        objContrato.Estatus = fila["Estatus"].ToString() == "True" ? "Activo" : "Inactivo";

                        lista.Add(objContrato);
                    }

                    return lista;
                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                throw new Exception("No se pudo obtener la información de los contratos");
            }
            finally
            {
                Conexion.desconectar();
            }
        }
        /// <summary>
        /// Método que Realiza consultas para obtener registros de la tabla Contratos.
        /// </summary>
        /// <param name="text">
        /// Parámetro de entrada que recibe una variable de tipo String que contiene una cadena
        /// de texto que se utilizara para buscar coincidencias con los registros de la tabla Contratos.
        /// </param>
        /// <returns>
        /// Regresa una lista de registros de la tabla Contratos que coincidan con la cadena de texto que
        /// se proporcionó.
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de que no se pudo realizar la conexión se mandara un mensaje donde indica que la
        /// conexión no se logró realizar.
        /// En caso de que no se encuentren datos en la tabla de Contratos mandara un mensaje que
```

```
/// indica que no se pudieron obtener dato de la tabla.
/// </exception>
public List<Contrato> Buscar(string text)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand(
                @"select * from contratos
                      where
                      idContrato like  CONCAT('%', @ref, '%') ||
                      precioServicio like  CONCAT('%', @ref, '%') ||
                      inicioContrato like  CONCAT('%', @ref, '%') ||
                      finContrato like  CONCAT('%', @ref, '%') ||
                      idCliente like  CONCAT('%', @ref, '%');"
            );
            comando.Parameters.AddWithValue("@ref", text);
            comando.Connection = Conexion.conexion;
            MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
            DataTable resultado = new DataTable();
            adapter.Fill(resultado);
            List<Contrato> lista = new List<Contrato>();
            Contrato objContrato = null;

            foreach (DataRow fila in resultado.Rows)
            {
                objContrato = new Contrato();
                objContrato.idContrato = fila["idContrato"].ToString();
                objContrato.precioServicio = double.Parse(fila["precioServicio"].ToString());
                objContrato.inicioContrato = fila["inicioContrato"].ToString();
                objContrato.finContrato = fila["finContrato"].ToString();
                objContrato.idCliente = fila["idCliente"].ToString();
                objContrato.Estatus = fila["Estatus"].ToString() == "True" ? "Activo" : "Inactivo";

                lista.Add(objContrato);
            }

            return lista;
        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        throw new Exception("No se pudo obtener la información de los contratos");
    }
    finally
    {
        Conexion.desconectar();
    }
}
/// <summary>
/// Método que realiza una consulta para obtener un registro en base a id proporcionado
/// </summary>
/// <param name="id">
/// Parámetro que recibe un variable de tipo String con la que se realizara una búsqueda
/// en la tabla de Contratos
/// </param>
/// <returns>
/// regresa un objeto de tipo Contrato que contendrá toda la información del registro
/// que coincida con el id proporcionado.
/// </returns>
/// <exception cref="Exception">
/// En caso de que no se pudo realizar la conexión se mandara un mensaje donde indica que la
/// conexión no se logró realizar.
/// En caso de que no se encuentren datos en la tabla de Contratos mandara un mensaje que
/// indica que no se pudieron obtener dato de la tabla.
/// </exception>
public Contrato ObtenerUno(string id)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand(@"SELECT * FROM contratos WHERE idContrato=@id");
            comando.Parameters.AddWithValue("@id", id);
            comando.Connection = Conexion.conexion;
            MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
            DataTable resultado = new DataTable();
            adapter.Fill(resultado);

            Contrato objContrato = null;
            if (resultado.Rows.Count > 0)
            {
                DataRow fila = resultado.Rows[0];
                objContrato = new Contrato();
                objContrato.idContrato = fila["idContrato"].ToString();
                objContrato.precioServicio = double.Parse(fila["precioServicio"].ToString());
                objContrato.inicioContrato = fila["inicioContrato"].ToString();
                objContrato.finContrato = fila["finContrato"].ToString();
```

```
                objContrato.idCliente = fila["idCliente"].ToString();
                Console.WriteLine(fila["Estatus"].ToString());
                objContrato.Estatus = fila["Estatus"].ToString() == "True" ? "Activo" : "Inactivo";
            }

            return objContrato;
        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        throw new Exception("No se pudo obtener la información del contrato");
    }
    finally
    {
        Conexion.desconectar();
    }
}

/// <summary>
/// Método que permite eliminar un registro de la tabla Contratos mediante un id proporcionado.
/// </summary>
/// <param name="id">
/// parámetro que recive una variable de tipo String que contiene un id para realizar un
/// consulta que coincida con el id proporcionado.
/// </param>
/// <returns>
/// Regresara una variable de tipo Boolean con el valor 1 o true si el registro se eliminó con éxito.
/// en caso de lo contrario regresara un valor 0 o false si no se logró eliminar el registro.
/// </returns>
/// <exception cref="Exception">
/// En caso de que no se pudo realizar la conexión se mandara un mensaje donde indica que la
/// conexión no se logró realizar.
/// En caso de que no se encuentre el registro o no se pueda eliminar mandara un mensaje donde indica
/// que no se pudo eliminar el registro.
/// </exception>
public bool eliminar(string id)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand(
                @"DELETE FROM contratos
                WHERE idContrato=@id");

            comando.Parameters.AddWithValue("@id", id);
            comando.Connection = Conexion.conexion;
            int filasBorradas = comando.ExecuteNonQuery();
            return (filasBorradas > 0);
        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        if (ex.Number == 1451)
        {
            MySqlCommand comando = new MySqlCommand(
                @"update contratos set Estatus=0
                WHERE idContrato=@id");

            comando.Parameters.AddWithValue("@id", id);
            comando.Connection = Conexion.conexion;
            int filasBorradas = comando.ExecuteNonQuery();
            return (filasBorradas > 0);
        }
        else
        {
            throw new Exception("Error al intentar eliminar el contrato");
        }

    }
    finally
    {
        Conexion.desconectar();
    }
}
/// <summary>
/// Método que permite Actualizar registros de la tabla Contratos.
/// </summary>
/// <param name="contrato">
/// parámetro de tipo Contrato que recibirá todos los datos del registro seleccionado para su actualización.
/// </param>
/// <returns>
/// Regresa una variable de tipo Boolean con un valor de 1 o true cuando el registro se pudo realizar con éxito,
/// en caso de lo contrario regresara un valor 0 o false.
/// </returns>
```

```
/// <exception cref="Exception">
/// En caso de que no se pudo realizar la conexión se mandara un mensaje donde indica que la
/// conexión no se logró realizar.
/// En caso de que la actualización no se pudo realizar mandara un mensaje que indique que no se pudo
/// realizar la actualización del registro.
/// </exception>
public bool editar(Contrato contrato)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand(
                @"UPDATE `dbwifimex`.`contratos` SET
                `precioServicio` = @cost,
                `inicioContrato` = @ini,
                `idCliente` = @idc,
                `finContrato` = @fin,
                `Estatus` = @estatus WHERE (`idContrato` = @id);"
            );
            comando.Parameters.AddWithValue("@id", contrato.idContrato);
            comando.Parameters.AddWithValue("@cost", contrato.precioServicio);
            comando.Parameters.AddWithValue("@ini", contrato.inicioContrato);
            comando.Parameters.AddWithValue("@fin", contrato.finContrato);
            comando.Parameters.AddWithValue("@idc", contrato.idCliente);
            comando.Parameters.AddWithValue("@estatus", contrato.Estatus == "Activo" ? 1 : 0);

            comando.Connection = Conexion.conexion;

            int filasBorradas = comando.ExecuteNonQuery();

            return (filasBorradas > 0);
        }
        else
        {
            return false;
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        throw new Exception("No se pudo editar la info del contrato");
    }
    finally
    {
        Conexion.desconectar();
    }
}
/// <summary>
/// Método que permite agregar registros a la tabla de Contratos.
/// </summary>
/// <param name="contrato">
/// parámetro tipo Contrato que recibe todos los datos del nuevo registro para su posterior
/// almacenamiento de registro.
/// </param>
/// <returns>
/// Regresa una variable de tipo int con el valor de 1 si el registro se realizó con éxito
/// y o en caso de que el registro no se logró realizar.
/// </returns>
/// <exception cref="Exception">
/// En caso de que no se pudo realizar la conexión se mandara un mensaje donde indica que la
/// conexión no se logró realizar.
/// En caso de que la inserción no se pudo realizar mandara un mensaje que indique que no se pudo
/// realizar la agregación del registro.
/// </exception>
public int agregar(Contrato contrato)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand(
                "INSERT INTO `dbwifimex`.`contratos` " +
                "(`idContrato`, `precioServicio`, `inicioContrato`, `finContrato`, `idCliente`, `Estatus`) VALUES " +
                "(@id,@cost , @ini, @fin, @idc, '1')");

            comando.Parameters.AddWithValue("@id", contrato.idContrato);
            comando.Parameters.AddWithValue("@cost", contrato.precioServicio);
            comando.Parameters.AddWithValue("@ini", contrato.inicioContrato);
            comando.Parameters.AddWithValue("@fin", contrato.finContrato);
            comando.Parameters.AddWithValue("@idc", contrato.idCliente);
            comando.Connection = Conexion.conexion;

            int filasAgregadas = comando.ExecuteNonQuery();

            return filasAgregadas;
        }
        else
        {
            return -1;
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
```

```
            catch (MySqlException ex)
            {
                throw new Exception("No se pudo agregar, intentelo de nuevo o comuniquese con el administrador");
            }
            finally
            {
                Conexion.desconectar();
            }
        }

    }
}
```

DAOEmpleados

```
using Modelos;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Datos
{
    /// <summary>
    /// Clase DAOEmpleados que contiene los métodos para realizar consultas, inserciones, actualizaciones y eliminación
    /// de registros de la tabla Empleados de la base de datos del programa.
    /// </summary>
    public class DAOEmpleados
    {
        /// <summary>
        /// Método que se utiliza para iniciar sección en el programa mediante la comparación
        /// de un usuario y contraseña en la base de datos.
        /// </summary>
        /// <param name="usuario">
        /// Parámetro que recibe un string con un usuario para su comparación en la base de datos
        /// </param>
        /// <param name="password">
        /// parámetro que recibe un sring con la contraseña para su comparación con una de la base de datos
        /// </param>
        /// <returns>
        /// Regresa un objeto de tipo usuario si se encontró al usuario y permite el acceso al programa
        /// en caso de lo contrario coincidencias no regresa un objeto vacío y no permite el acceso al programa.
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// </exception>
        public Empleados IniciarSeccion(string usuario, string password)
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand("SELECT * FROM empleados WHERE idEmpleado=@Usuario AND
pas=sha1(@Password) and Estatus=1");
                    comando.Parameters.AddWithValue("@Usuario", usuario);
                    comando.Parameters.AddWithValue("@Password", password);
                    comando.Connection = Conexion.conexion;
                    MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                    DataTable resultado = new DataTable();
                    adapter.Fill(resultado);
                    Empleados objUsuario = null;
                    if (resultado.Rows.Count > 0 && resultado.Rows.Count == 1)
                    {
                        objUsuario = new Empleados(
                            resultado.Rows[0]["idEmpleado"].ToString(),
                            resultado.Rows[0]["nomEmpleados"].ToString()
                            );
                    }
                    return objUsuario;
                }
                else
                {
                    throw new Exception("No se a podido conectar con el servidor");
                }
            }
            finally
            {
                Conexion.desconectar();
            }
        }
        /// <summary>
        /// Metodo que obtiene un lista con todos los registros de la tabla de Empleados.
        /// </summary>
        /// <returns>
        /// Regresa una lista de Tipo Empleados con todos los registros de la tabla Empelados
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
        /// registros en la tabla
        /// </exception>
        public List<Empleados> ObtenerEmpleados()
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand("select * from empleados");
                    comando.Connection = Conexion.conexion;
                    MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
```

```
                DataTable resultado = new DataTable();
                adapter.Fill(resultado);
                List<Empleados> lista = new List<Empleados>();
                Empleados objEmpleado = null;
                foreach (DataRow fila in resultado.Rows)
                {
                    objEmpleado = new Empleados();
                    objEmpleado.Idempleado = (fila["idEmpleado"].ToString());
                    objEmpleado.Nombrecompleto = fila["nomEmpleados"].ToString();
                    objEmpleado.RFC = fila["RFC"].ToString();
                    objEmpleado.CURP = fila["CURP"].ToString();
                    objEmpleado.Direccion = fila["Direccion"].ToString();
                    objEmpleado.Telefono = fila["Telefono"].ToString();
                    objEmpleado.Correo = fila["Correo"].ToString();
                    objEmpleado.fechaContratacion = fila["fechaContratacion"].ToString();
                    objEmpleado.Rol = fila["Rol"].ToString();
                    objEmpleado.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);
                    lista.Add(objEmpleado);
                }

                return lista;
            }
            else
            {
                throw new Exception("No se ha podido conectar con el servidor");
            }
        }
        catch (MySqlException ex)
        {
            throw new Exception("No se pudo obtener la información de los empleados");
        }
        finally
        {
            Conexion.desconectar();
        }
    }
    /// <summary>
    /// Método que obtiene un lista con todos los registros de la tabla de Empleados que coincidan con
    /// el ID propocionado.
    /// </summary>
    /// <param name="id">
    /// Parametro que recive un String con ID para buscar un registro en la tabla de Empleados
    /// </param>
    /// <returns>
    /// Regresa una lista de Tipo Empleados con todos los registros de la tabla Empelados que coincidan
    /// con el ID proporcionado.
    /// </returns>
    /// <exception cref="Exception">
    /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
    /// conectar al servidor.
    /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
    /// registros en la tabla
    /// </exception>
    public List<Empleados> ObtenerEmpleado(string id)
    {
        try
        {
            if (Conexion.conectar())
            {
                MySqlCommand comando = new MySqlCommand("select * from empleados where idEmpleado=@id");
                comando.Parameters.AddWithValue("@id", id);
                comando.Connection = Conexion.conexion;
                MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                DataTable resultado = new DataTable();
                adapter.Fill(resultado);
                List<Empleados> lista = new List<Empleados>();
                Empleados objEmpleado = null;
                foreach (DataRow fila in resultado.Rows)
                {
                    objEmpleado = new Empleados();
                    objEmpleado.Idempleado = (fila["idEmpleado"].ToString());
                    objEmpleado.Nombrecompleto = fila["nomEmpleados"].ToString();
                    objEmpleado.RFC = fila["RFC"].ToString();
                    objEmpleado.CURP = fila["CURP"].ToString();
                    objEmpleado.Direccion = fila["Direccion"].ToString();
                    objEmpleado.Telefono = fila["Telefono"].ToString();
                    objEmpleado.Correo = fila["Correo"].ToString();
                    objEmpleado.fechaContratacion = fila["fechaContratacion"].ToString();
                    objEmpleado.Rol = fila["Rol"].ToString();
                    objEmpleado.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);
                    lista.Add(objEmpleado);
                }

                return lista;
            }
            else
            {
                throw new Exception("No se ha podido conectar con el servidor");
            }
        }
        catch (MySqlException ex)
        {
            throw new Exception("No se pudo obtener la información de los empleados");
```

```
        }
        finally
        {
            Conexion.desconectar();
        }
    }

    /// <summary>
    /// Método que obtiene un lista con todos los registros de la tabla de Empleados que están activos.
    /// </summary>
    /// <returns>
    /// Regresa una lista de Tipo Empleados con todos los registros de la tabla Empelados que estén
    /// en estado activo
    /// </returns>
    /// <exception cref="Exception">
    /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
    /// conectar al servidor.
    /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
    /// registros en la tabla
    /// </exception>
    public List<Empleados> ObtenerEmpleadosActivos()
    {
        try
        {
            if (Conexion.conectar())
            {
                MySqlCommand comando = new MySqlCommand("select * from empleados where Estatus=true");
                comando.Connection = Conexion.conexion;
                MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                DataTable resultado = new DataTable();
                adapter.Fill(resultado);
                List<Empleados> lista = new List<Empleados>();
                Empleados objEmpleado = null;
                foreach (DataRow fila in resultado.Rows)
                {
                    objEmpleado = new Empleados();
                    objEmpleado.Idempleado = (fila["idEmpleado"].ToString());
                    objEmpleado.Nombrecompleto = fila["nomEmpleados"].ToString();
                    objEmpleado.RFC = fila["RFC"].ToString();
                    objEmpleado.CURP = fila["CURP"].ToString();
                    objEmpleado.Direccion = fila["Direccion"].ToString();
                    objEmpleado.Telefono = fila["Telefono"].ToString();
                    objEmpleado.Correo = fila["Correo"].ToString();
                    objEmpleado.fechaContratacion = fila["fechaContratacion"].ToString();
                    objEmpleado.Rol = fila["Rol"].ToString();
                    objEmpleado.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);
                    lista.Add(objEmpleado);
                }

                return lista;
            }
            else
            {
                throw new Exception("No se ha podido conectar con el servidor");
            }
        }
        catch (MySqlException ex)
        {
            throw new Exception("No se pudo obtener la información de los empleados");
        }
        finally
        {
            Conexion.desconectar();
        }
    }
    /// <summary>
    /// Método que obtiene una lista con todos los registros de la tabla de Empleados que están inactivos.
    /// </summary>
    /// <returns>
    /// Regresa una lista de Tipo Empleados con todos los registros de la tabla Empelados que estén
    /// en estado inactivo
    /// </returns>
    /// <exception cref="Exception">
    /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
    /// conectar al servidor.
    /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
    /// registros en la tabla
    /// </exception>
    public List<Empleados> ObtenerEmpleadosInactivos()
    {
        try
        {
            if (Conexion.conectar())
            {
                MySqlCommand comando = new MySqlCommand("select * from empleados where Estatus=false");
                comando.Connection = Conexion.conexion;
                MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                DataTable resultado = new DataTable();
                adapter.Fill(resultado);
                List<Empleados> lista = new List<Empleados>();
                Empleados objEmpleado = null;
                foreach (DataRow fila in resultado.Rows)
                {
```

```
                objEmpleado = new Empleados();
                objEmpleado.Idempleado = (fila["idEmpleado"].ToString());
                objEmpleado.Nombrecompleto = fila["nomEmpleados"].ToString();
                objEmpleado.RFC = fila["RFC"].ToString();
                objEmpleado.CURP = fila["CURP"].ToString();
                objEmpleado.Direccion = fila["Direccion"].ToString();
                objEmpleado.Telefono = fila["Telefono"].ToString();
                objEmpleado.Correo = fila["Correo"].ToString();
                objEmpleado.fechaContratacion = fila["fechaContratacion"].ToString();
                objEmpleado.Rol = fila["Rol"].ToString();
                objEmpleado.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);
                lista.Add(objEmpleado);
            }

            return lista;
        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        throw new Exception("No se pudo obtener la información de los empleados");
    }
    finally
    {
        Conexion.desconectar();
    }
}
/// <summary>
/// Método que obtiene un objeto con el registro de la tabla de Empleados que coincidan con
/// el ID proporcionado.
/// </summary>
/// <param name="id">
/// parámetro que recibe un String con ID para buscar un registro en la tabla de Empleados
/// </param>
/// <returns>
/// Regresa un objeto de Tipo Empleados con todos los registros de la tabla Empelados que coincidan
/// con el ID proporcionado.
/// </returns>
/// <exception cref="Exception">
/// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
/// conectar al servidor.
/// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
/// registros en la tabla
/// </exception>
public Empleados ObtenerUnEmpleado(string id)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand("select * from empleados where idEmpleado=@id");
            comando.Parameters.AddWithValue("@id", id);
            comando.Connection = Conexion.conexion;
            MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
            DataTable resultado = new DataTable();
            adapter.Fill(resultado);
            Empleados objEmpleado = null;
            if (resultado.Rows.Count > 0)
            {
                objEmpleado = new Empleados();
                DataRow fila = resultado.Rows[0];
                objEmpleado.Idempleado = fila["idEmpleado"].ToString();
                objEmpleado.Nombrecompleto = fila["nomEmpleados"].ToString();
                objEmpleado.RFC = fila["RFC"].ToString();
                objEmpleado.CURP = fila["CURP"].ToString();
                objEmpleado.Edad = Convert.ToInt32(fila["Edad"]);
                objEmpleado.Direccion = fila["Direccion"].ToString();
                objEmpleado.Telefono = fila["Telefono"].ToString();
                objEmpleado.Correo = fila["Correo"].ToString();
                objEmpleado.fechaContratacion = fila["fechaContratacion"].ToString();
                objEmpleado.Rol = fila["Rol"].ToString();
            }

            return objEmpleado;
        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        throw new Exception("No se pudo obtener la informacion del empleado");
    }
    finally
    {
        Conexion.desconectar();
    }
}
```

```
        /// <summary>
        /// Método para realizar un nuevo registro en la tabla de empleados.
        /// </summary>
        /// <param name="emp">
        /// objeto de tipo empleado que recibirá todos los datos para el nuevo registro
        /// de la tabla empleados.
        /// </param>
        /// <returns>
        /// Regresa una int con un valor de 1 si el registro se pudo realizar y un o si no se realizó el registro
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de que no se el registro no se pudo realizar mandara un mensaje donde indica
        /// que no se pudo realizar el registro.
        /// </exception>
        public int AgregarEmpleado(Empleados emp)
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand(
                    "INSERT INTO empleados
VALUES(@idEmpleado,@name,@RFC,@CURP,@Edad,@direccion,@telefono,@correo,@fechacontratacion,@rol,sha1(@password),@State);");
                    comando.Parameters.AddWithValue("@idEmpleado", emp.Idempleado);
                    comando.Parameters.AddWithValue("@name", emp.Nombrecompleto);
                    comando.Parameters.AddWithValue("@RFC", emp.RFC);
                    comando.Parameters.AddWithValue("@CURP", emp.CURP);
                    comando.Parameters.AddWithValue("@Edad", Convert.ToInt32(emp.Edad));
                    comando.Parameters.AddWithValue("@direccion", emp.Direccion);
                    comando.Parameters.AddWithValue("@telefono", emp.Telefono);
                    comando.Parameters.AddWithValue("@correo", emp.Correo);
                    comando.Parameters.AddWithValue("@fechacontratacion", emp.fechaContratacion);
                    comando.Parameters.AddWithValue("@rol", emp.Rol);
                    comando.Parameters.AddWithValue("@password", emp.Password);
                    comando.Parameters.AddWithValue("@State", Convert.ToInt32(emp.Estatus));
                    comando.Connection = Conexion.conexion;
                    int filasAgregadas = Convert.ToInt32(comando.ExecuteScalar());
                    return filasAgregadas;
                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                if (ex.Number == 1062)
                {
                    MySqlCommand comando = new MySqlCommand(
                        @"update Empleados set Estatus=true where idEmpleado=@idEmpleado and Estatus=false");
                    comando.Parameters.AddWithValue("@idEmpeado", emp.Idempleado);
                    comando.Connection = Conexion.conexion;
                    comando.Connection = Conexion.conexion;
                    return comando.ExecuteNonQuery();
                }
                else
                {
                    throw new Exception("No se ha podido agregar el empleado");
                }
            }
            finally
            {
                Conexion.desconectar();
            }

        }
        /// <summary>
        /// Metodo para actualizar un registro de la tabla de empleados
        /// </summary>
        /// <param name="emp">
        /// objeto de tipo empleado que recibirá todos los datos para la actualización
        /// del registro de la tabla empleados.
        /// </param>
        /// <returns>
        /// Regresa una variable de tipo Boolean con valor 1 0 true si el registro se pudo actualizar,
        /// y un valor de 0 o false si no se pudo actualizar el registro
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de que no se el registro no se pudo actualizar mandara un mensaje donde indica
        /// que no se pudo realizar el registro.
        /// </exception>
        public bool ModificarEmpleado(Empleados emp)
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand(
```

```
                    "update empleados set
nomEmpleados=@name,RFC=@RFC,CURP=@CURP,Edad=@Edad,Direccion=@direccion,Telefono=@telefono," +
                    "Correo=@correo,Rol=@rol, Estatus=true where idEmpleado=@idEmpleado");
                    comando.Parameters.AddWithValue("@idEmpleado", emp.Idempleado);
                    comando.Parameters.AddWithValue("@name", emp.Nombrecompleto);
                    comando.Parameters.AddWithValue("@RFC", emp.RFC);
                    comando.Parameters.AddWithValue("@CURP", emp.CURP);
                    comando.Parameters.AddWithValue("@Edad", Convert.ToInt32(emp.Edad));
                    comando.Parameters.AddWithValue("@direccion", emp.Direccion);
                    comando.Parameters.AddWithValue("@telefono", emp.Telefono);
                    comando.Parameters.AddWithValue("@correo", emp.Correo);
                    comando.Parameters.AddWithValue("@rol", emp.Rol);
                    comando.Connection = Conexion.conexion;
                    int filasEditadas = comando.ExecuteNonQuery();
                    return (filasEditadas > 0);
                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                throw new Exception("No se pudo actualizar la información del empleado");
            }
            finally
            {
                Conexion.desconectar();
            }
        }
        /// <summary>
        /// Metodo que elimina un registro de la tabla de empleados
        /// </summary>
        /// <param name="id">
        /// parámetro que recibe un String con un ID proporcionado para buscar un registro
        /// para su posterior eliminación.
        /// </param>
        /// <returns>
        /// Regresa una variable de tipo Boolean con valor 1 0 true si el registro se pudo eliminar,
        /// y un valor de 0 o false si no se pudo eliminar el registro
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de que no el registro no se pudo eliminar mandara un mensaje donde indica
        /// que no se pudo realizar el registro.
        /// </exception>
        public bool EliminarEmpleado(string id)
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand(
                        @"delete from empleados where idEmpleado=@Id");
                    comando.Parameters.AddWithValue("@id", id);
                    comando.Connection = Conexion.conexion;
                    int filasBorradas = comando.ExecuteNonQuery();
                    return (filasBorradas > 0);

                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                if (ex.Number == 1451)
                {
                    try
                    {
                        MySqlCommand comando = new MySqlCommand(
                        @"update empleados set Estatus=false where idEmpleado=@Id");
                        comando.Parameters.AddWithValue("@id", id);
                        comando.Connection = Conexion.conexion;
                        int filasBorradas = comando.ExecuteNonQuery();
                        return (filasBorradas > 0);
                    }
                    catch
                    {
                        throw new Exception("Usuario ya eliminado");
                    }
                }
                else
                {
                    throw new Exception("Error al intentar eliminar el empleado");
                }

            }
            finally
            {
                Conexion.desconectar();
```

```
            }
        }
    }
}
```

DAOInstalacion

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Modelos;
using MySql.Data.MySqlClient;
using System.Data;

namespace Datos
{
    /// <summary>
    /// Clase DAOIntalaciones que contiene los métodos para realizar consultas, inserciones y actualizaciones
    /// de registros de la tabla Instalaciones de la base de datos del programa.
    /// </summary>
    public class DAOInstalaciones
    {
        /// <summary>
        /// Método que obtiene una lista con todos los registros de la tabla de Instalaciones.
        /// </summary>
        /// <returns>
        /// Regresa una lista de Tipo Instalaciones con todos los registros de la tabla Instalaciones.
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
        /// registros en la tabla
        /// </exception>
        public List<Instalaciones> ObtenerInstalaciones()
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand("select * from instalaciones");
                    comando.Connection = Conexion.conexion;
                    MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                    DataTable resultado = new DataTable();
                    adapter.Fill(resultado);
                    List<Instalaciones> lista = new List<Instalaciones>();
                    Instalaciones objInstalacion = null;
                    foreach (DataRow fila in resultado.Rows)
                    {
                        objInstalacion = new Instalaciones();
                        objInstalacion.IdInstalacion = (fila["idInstalacion"].ToString());
                        objInstalacion.fechaInstalacin = fila["fechaInstalacion"].ToString();
                        objInstalacion.idEmpelado = fila["idEmpleado"].ToString();
                        objInstalacion.idContratos = fila["idContrato"].ToString();
                        objInstalacion.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);
                        lista.Add(objInstalacion);
                    }

                    return lista;
                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                throw new Exception("No se pudo obtener la información de la instalacion");
            }
            finally
            {
                Conexion.desconectar();
            }
        }
        /// <summary>
        /// Método que obtiene una lista con todos los registros de la tabla de Instalaciones que están activos.
        /// </summary>
        /// <returns>
        /// Regresa una lista de Tipo Instalaciones con todos los registros de la tabla Instalaciones que estén
        /// en estado activo
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
        /// registros en la tabla
        /// </exception>
        public List<Instalaciones> ObtenerInstalacionesActivos()
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand("select * from instalaciones where Estatus=true");
                    comando.Connection = Conexion.conexion;
```

```
                MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                DataTable resultado = new DataTable();
                adapter.Fill(resultado);
                List<Instalaciones> lista = new List<Instalaciones>();
                Instalaciones objInstalacion = null;
                foreach (DataRow fila in resultado.Rows)
                {
                    objInstalacion = new Instalaciones();
                    objInstalacion.IdInstalacion = (fila["idInstalacion"].ToString());
                    objInstalacion.fechaInstalacin = fila["fechaInstalacion"].ToString();
                    objInstalacion.idEmpelado = fila["idEmpleado"].ToString();
                    objInstalacion.idContratos = fila["idContrato"].ToString();
                    objInstalacion.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);
                    lista.Add(objInstalacion);
                }

                return lista;
            }
            else
            {
                throw new Exception("No se ha podido conectar con el servidor");
            }
        }
        catch (MySqlException ex)
        {
            throw new Exception("No se pudo obtener la información de la instalacion");
        }
        finally
        {
            Conexion.desconectar();
        }
    }
    /// <summary>
    /// Método que obtiene una lista con todos los registros de la tabla de Instalaciones que están inactivos.
    /// </summary>
    /// <returns>
    /// Regresa una lista de Tipo Instalaciones con todos los registros de la tabla Instalaciones que estén
    /// en estado inactivo
    /// </returns>
    /// <exception cref="Exception">
    /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
    /// conectar al servidor.
    /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
    /// registros en la tabla
    /// </exception>
    public List<Instalaciones> ObtenerInstalacionesInactivos()
    {
        try
        {
            if (Conexion.conectar())
            {
                MySqlCommand comando = new MySqlCommand("select * from instalaciones where Estatus=false");
                comando.Connection = Conexion.conexion;
                MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                DataTable resultado = new DataTable();
                adapter.Fill(resultado);
                List<Instalaciones> lista = new List<Instalaciones>();
                Instalaciones objInstalacion = null;
                foreach (DataRow fila in resultado.Rows)
                {
                    objInstalacion = new Instalaciones();
                    objInstalacion.IdInstalacion = (fila["idInstalacion"].ToString());
                    objInstalacion.fechaInstalacin = fila["fechaInstalacion"].ToString();
                    objInstalacion.idEmpelado = fila["idEmpleado"].ToString();
                    objInstalacion.idContratos = fila["idContrato"].ToString();
                    objInstalacion.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);
                    lista.Add(objInstalacion);
                }

                return lista;
            }
            else
            {
                throw new Exception("No se ha podido conectar con el servidor");
            }
        }
        catch (MySqlException ex)
        {
            throw new Exception("No se pudo obtener la información de la instalacion");
        }
        finally
        {
            Conexion.desconectar();
        }
    }

    /// <summary>
    /// Método que obtiene una lista con todos los registros de la tabla de Instalaciones que coincidan con
    /// el ID propocionado.
    /// </summary>
    /// <param name="id">
    /// parámetro que recibe un String con ID para buscar un registro en la tabla de Instalaciones
    /// </param>
```

```
        /// <returns>
        /// Regresa una lista de Tipo Instalaciones con todos los registros de la tabla Instalaciones que coincidan
        /// con el ID proporcionado.
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
        /// registros en la tabla
        /// </exception>
        public List<Instalaciones> ObtenerInstalacion(string id)
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand("select * from instalaciones where idInstalacion=@id or
idEmpleado=@id or idContrato=@id");
                    comando.Parameters.AddWithValue("@id", id);
                    comando.Connection = Conexion.conexion;
                    MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                    DataTable resultado = new DataTable();
                    adapter.Fill(resultado);
                    List<Instalaciones> lista = new List<Instalaciones>();
                    Instalaciones objInstalacion = null;
                    foreach (DataRow fila in resultado.Rows)
                    {
                        objInstalacion = new Instalaciones();
                        objInstalacion.IdInstalacion = (fila["idInstalacion"].ToString());
                        objInstalacion.fechaInstalacin = fila["fechaInstalacion"].ToString();
                        objInstalacion.idEmpelado = fila["idEmpleado"].ToString();
                        objInstalacion.idContratos = fila["idContrato"].ToString();
                        objInstalacion.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);
                        lista.Add(objInstalacion);
                    }

                    return lista;
                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                throw new Exception("No se pudo obtener la información de la instalacion");
            }
            finally
            {
                Conexion.desconectar();
            }
        }
        /// <summary>
        /// Método que obtiene un objeto con el registro de la tabla de Instalaciones que coincidan con
        /// el ID propocionado.
        /// </summary>
        /// <param name="id">
        /// parámetro que recibe un String con ID para buscar un registro en la tabla de Instalaciones
        /// </param>
        /// <returns>
        /// Regresa un objeto de Tipo Instalaciones con todos los registros de la tabla Instalaciones que coincidan
        /// con el ID proporcionado.
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
        /// registros en la tabla
        /// </exception>
        public Instalaciones ObtenerUnaInstalacion(string id)
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand("select * from instalaciones where idInstalacion=@id");
                    comando.Parameters.AddWithValue("@id", id);
                    comando.Connection = Conexion.conexion;
                    MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                    DataTable resultado = new DataTable();
                    adapter.Fill(resultado);
                    Instalaciones objInstalacion = null;
                    if (resultado.Rows.Count > 0)
                    {
                        objInstalacion = new Instalaciones();
                        DataRow fila = resultado.Rows[0];
                        objInstalacion = new Instalaciones();
                        objInstalacion.IdInstalacion = (fila["idInstalacion"].ToString());
                        objInstalacion.fechaInstalacin = fila["fechaInstalacion"].ToString();
                        objInstalacion.idEmpelado = fila["idEmpleado"].ToString();
                        objInstalacion.idContratos = fila["idContrato"].ToString();
                    }
```

```
            return objInstalacion;
        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        throw new Exception("No se pudo obtener la informacion de la instalacion");
    }
    finally
    {
        Conexion.desconectar();
    }
}

/// <summary>
/// Método para realizar un nuevo registro en la tabla de Instalaciones.
/// </summary>
/// <param name="ins">
/// objeto de tipo Instalaciones que recibirá todos los datos para el nuevo registro
/// de la tabla Instalaciones.
/// </param>
/// <returns>
/// Regresa una int con un valor de 1 si el registro se pudo realizar y un o si no se realizó el registro
/// </returns>
/// <exception cref="Exception">
/// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
/// conectar al servidor.
/// En caso de que no se el registro no se pudo realizar mandara un mensaje donde indica
/// que no se pudo realizar el registro.
/// </exception>
public int AgregarInstlacion(Instalaciones ins)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand(
            "INSERT INTO instalaciones VALUES(@idInstalacion,@fechaInstalacion,@idEmpleado,@idContrato,@State);");
            comando.Parameters.AddWithValue("@idInstalacion", ins.IdInstalacion);
            comando.Parameters.AddWithValue("@fechaInstalacion", ins.fechaInstalacin);
            comando.Parameters.AddWithValue("@idEmpleado", ins.idEmpelado);
            comando.Parameters.AddWithValue("@idContrato", ins.idContratos);
            comando.Parameters.AddWithValue("@State", Convert.ToInt32(ins.Estatus));
            comando.Connection = Conexion.conexion;
            int filasAgregadas = Convert.ToInt32(comando.ExecuteScalar());
            return filasAgregadas;
        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        if (ex.Number == 1062)
        {
            MySqlCommand comando = new MySqlCommand(
                @"update instalaciones set Estatus=true where idInstalacion=@idInstalacion and Estatus=false");
            comando.Parameters.AddWithValue("@idInstalacion", ins.IdInstalacion);
            comando.Connection = Conexion.conexion;
            comando.Connection = Conexion.conexion;
            return comando.ExecuteNonQuery();
        }
        else
        {
            throw new Exception("No se ha podido agregar la instalacion");
        }
    }
    finally
    {
        Conexion.desconectar();
    }

}

/// <summary>
/// Método para actualizar un registro de la tabla de Instalaciones
/// </summary>
/// <param name="ins">
/// objeto de tipo Instalaciones que recibirá todos los datos para la actualización
/// del registro de la tabla empleados.
/// </param>
/// <returns>
/// Regresa una variable de tipo Boolean con valor 1 0 true si el registro se pudo actualizar,
/// y un valor de 0 o false si no se pudo actualizar el registro
/// </returns>
/// <exception cref="Exception">
/// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
/// conectar al servidor.
/// En caso de que no se el registro no se pudo actualizar mandara un mensaje donde indica
```

```
        /// que no se pudo realizar el registro.
        /// </exception>
        public bool ModificarInstalacion(Instalaciones ins)
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand(
                    "update instalaciones set idInstalacion=@idInstalacion,fechaInstalacion=@fechaInstalacion," +
                    "idEmpleado=@idEmpleado,idContrato=@idContrato where idInstalacion=@idInstalacion");
                    comando.Parameters.AddWithValue("@idInstalacion", ins.IdInstalacion);
                    comando.Parameters.AddWithValue("@fechaInstalacion", ins.fechaInstalacin);
                    comando.Parameters.AddWithValue("@idEmpleado", ins.idEmpelado);
                    comando.Parameters.AddWithValue("@idContrato", ins.idContratos);
                    comando.Connection = Conexion.conexion;
                    int filasEditadas = comando.ExecuteNonQuery();
                    return (filasEditadas > 0);
                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                throw new Exception("No se pudo actualizar la información de la instalacion");
            }
            finally
            {
                Conexion.desconectar();
            }
        }


    }
}
```

DAOProductos

```
using Modelos;
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Datos
{
    /// <summary>
    /// Clase DAOProductos que contiene los métodos para realizar consultas, inserciones, actualizaciones y eliminación
    /// de registros de la tabla Empleados de la base de datos del programa.
    /// </summary>
    public class DAOProductos
    {
        /// <summary>
        /// Método que obtiene una lista con todos los registros de la tabla de Productos.
        /// </summary>
        /// <returns>
        /// Regresa una lista de Tipo Productos con todos los registros de la tabla Productos
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
        /// registros en la tabla
        /// </exception>
        public List<Productos> ObtenerProductos()
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand("select * from productos");
                    comando.Connection = Conexion.conexion;
                    MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                    DataTable resultado = new DataTable();
                    adapter.Fill(resultado);
                    List<Productos> lista = new List<Productos>();
                    Productos objProductos = null;
                    foreach (DataRow fila in resultado.Rows)
                    {
                        objProductos = new Productos();
                        objProductos.codigoBarra = fila["codigoBarra"].ToString();
                        objProductos.nomProducto = fila["nomProducto"].ToString();
                        objProductos.fechaRegistro = fila["fechaRegistro"].ToString();
                        objProductos.idProveedores = fila["idProveedores"].ToString();
                        objProductos.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);
                        lista.Add(objProductos);
                    }

                    return lista;
                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                throw new Exception("No se pudo obtener la información de los productos");
            }
            finally
            {
                Conexion.desconectar();
            }
        }

        /// <summary>
        /// Método que obtiene una lista con todos los registros de la tabla de Productos que están activos.
        /// </summary>
        /// <returns>
        /// Regresa una lista de Tipo Productos con todos los registros de la tabla Productos que Esten
        /// en estado activo
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
        /// registros en la tabla
        /// </exception>
        public List<Productos> ObtenerProductosActivos()
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand("select * from productos where Estatus=true");
```

```
                comando.Connection = Conexion.conexion;
                MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                DataTable resultado = new DataTable();
                adapter.Fill(resultado);
                List<Productos> lista = new List<Productos>();
                Productos objProductos = null;
                foreach (DataRow fila in resultado.Rows)
                {
                    objProductos = new Productos();
                    objProductos.codigoBarra = fila["codigoBarra"].ToString();
                    objProductos.nomProducto = fila["nomProducto"].ToString();
                    objProductos.fechaRegistro = fila["fechaRegistro"].ToString();
                    objProductos.idProveedores = fila["idProveedores"].ToString();
                    objProductos.nomProveedor = this.ObtenerUnProveedor(objProductos.idProveedores).nomProveedores;
                    objProductos.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);
                    lista.Add(objProductos);
                }

                return lista;
            }
            else
            {
                throw new Exception("No se ha podido conectar con el servidor");
            }
        }
        catch (MySqlException ex)
        {
            throw new Exception("No se pudo obtener la información de los productos");
        }
        finally
        {
            Conexion.desconectar();
        }
    }
    /// <summary>
    /// Método que obtiene una lista con todos los registros de la tabla de Productos que están inactivos.
    /// </summary>
    /// <returns>
    /// Regresa una lista de Tipo Productos con todos los registros de la tabla Productos que Esten
    /// en estado inactivo
    /// </returns>
    /// <exception cref="Exception">
    /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
    /// conectar al servidor.
    /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
    /// registros en la tabla
    /// </exception>
    public List<Productos> ObtenerProductosInactivos()
    {
        try
        {
            if (Conexion.conectar())
            {
                MySqlCommand comando = new MySqlCommand("select * from productos where Estatus=false");
                comando.Connection = Conexion.conexion;
                MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                DataTable resultado = new DataTable();
                adapter.Fill(resultado);
                List<Productos> lista = new List<Productos>();
                Productos objProductos = null;
                foreach (DataRow fila in resultado.Rows)
                {
                    objProductos = new Productos();
                    objProductos.codigoBarra = fila["codigoBarra"].ToString();
                    objProductos.nomProducto = fila["nomProducto"].ToString();
                    objProductos.fechaRegistro = fila["fechaRegistro"].ToString();
                    objProductos.idProveedores = fila["idProveedores"].ToString();
                    objProductos.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);
                    lista.Add(objProductos);
                }

                return lista;
            }
            else
            {
                throw new Exception("No se ha podido conectar con el servidor");
            }
        }
        catch (MySqlException ex)
        {
            throw new Exception("No se pudo obtener la información de los productos");
        }
        finally
        {
            Conexion.desconectar();
        }
    }

    /// <summary>
    /// Método que obtiene un objeto con el registro de la tabla de Productos que coincidan con
    /// el ID proporcionado.
    /// </summary>
    /// <param name="id">
```

```
        /// parámetro que recibe un String con ID para buscar un registro en la tabla de Proveedores
        /// </param>
        /// <returns>
        /// Regresa un objeto de Tipo Productos con todos los registros de la tabla Productos que coincidan
        /// con el ID proporcionado.
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
        /// registros en la tabla
        /// </exception>
        public Productos ObtenerUnProducto(string codigo)
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand("select * from productos where codigoBarra=@codigo");
                    comando.Parameters.AddWithValue("@codigo", codigo);
                    comando.Connection = Conexion.conexion;
                    MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                    DataTable resultado = new DataTable();
                    adapter.Fill(resultado);
                    Productos objProductos = null;
                    if (resultado.Rows.Count > 0)
                    {
                        objProductos = new Productos();
                        DataRow fila = resultado.Rows[0];
                        objProductos = new Productos();

                        objProductos.codigoBarra = fila["codigoBarra"].ToString();
                        objProductos.nomProducto = (fila["nomProducto"].ToString());
                        objProductos.fechaRegistro = fila["fechaRegistro"].ToString();
                        objProductos.idProveedores = fila["idProveedores"].ToString();
                        objProductos.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);
                    }

                    return objProductos;
                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                throw new Exception("No se pudo obtener la informacion del producto");
            }
            finally
            {
                Conexion.desconectar();
            }
        }

        /// <summary>
        /// Método que obtiene una lista con todos los registros de la tabla de Proveedores que están activos.
        /// </summary>
        /// <returns>
        /// Regresa una lista de Tipo Proveedores con todos los registros de la tabla Proveedores que estén
        /// en estado activo
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
        /// registros en la tabla
        /// </exception>
        public List<Proveedores> ObtenerProveedoresActivos()
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand("select idProveedores, nomProveedor from proveedores where
Estatus=true");
                    comando.Connection = Conexion.conexion;
                    MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                    DataTable resultado = new DataTable();
                    adapter.Fill(resultado);
                    List<Proveedores> lista = new List<Proveedores>();
                    Proveedores objProveedores = null;
                    foreach (DataRow fila in resultado.Rows)
                    {
                        objProveedores = new Proveedores();
                        objProveedores.idProveedores = fila["idProveedores"].ToString();
                        objProveedores.nomProveedores = fila["nomProveedor"].ToString();
                        objProveedores.RFC = null;
                        objProveedores.Direccion = null;
                        objProveedores.Telefono = null;
                        objProveedores.Correo = null;
                        objProveedores.fechaRegistro = null;
                        objProveedores.Estatus = true;
```

```
                lista.Add(objProveedores);
            }

            return lista;
        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        throw new Exception("No se pudo obtener la información de los proveedores");
    }
    finally
    {
        Conexion.desconectar();
    }
}

/// <summary>
/// Metodo que obtiene un objeto con el registro de la tabla de Proveedores que coincidan con
/// el ID proporcionado.
/// </summary>
/// <param name="id">
/// parámetro que recibe un String con ID para buscar un registro en la tabla de Proveedores
/// </param>
/// <returns>
/// Regresa un objeto de Tipo Proveedores con todos los registros de la tabla Proveedores que coincidan
/// con el ID proporcionado.
/// </returns>
/// <exception cref="Exception">
/// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
/// conectar al servidor.
/// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
/// registros en la tabla
/// </exception>
public Proveedores ObtenerUnProveedor(string id)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand("select nomProveedor from proveedores where idProveedores=@id");
            comando.Parameters.AddWithValue("@id", id);
            comando.Connection = Conexion.conexion;
            MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
            DataTable resultado = new DataTable();
            adapter.Fill(resultado);
            Proveedores objProveedores = null;
            if (resultado.Rows.Count > 0)
            {
                objProveedores = new Proveedores();
                DataRow fila = resultado.Rows[0];
                objProveedores = new Proveedores();

                objProveedores.idProveedores = null;
                objProveedores.nomProveedores = (fila["nomProveedor"].ToString());
                objProveedores.RFC = null;
                objProveedores.Direccion = null;
                objProveedores.Telefono = null;
                objProveedores.Correo = null;
                objProveedores.fechaRegistro = null;
                objProveedores.Estatus = true;
            }

            return objProveedores;
        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        throw new Exception("No se pudo obtener la informacion del proveedor");
    }
    finally
    {
        Conexion.desconectar();
    }
}

/// <summary>
/// Método para realizar un nuevo registro en la tabla de Productos.
/// </summary>
/// <param name="ins">
/// objeto de tipo Productos que recibirá todos los datos para el nuevo registro
/// de la tabla Productos.
/// </param>
/// <returns>
/// Regresa una int con un valor de 1 si el registro se pudo realizar y un o si no se realizó el registro
/// </returns>
```

```
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de que no se el registro no se pudo realizar mandara un mensaje donde indica
        /// que no se pudo realizar el registro.
        /// </exception>
        public int AgregarProducto(Productos ins)
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand(
                    "INSERT INTO productos VALUES(@codigoBarra, @nomProducto, @fechaRegistro, @idProveedores, @Estatus);");
                    comando.Parameters.AddWithValue("@codigoBarra", ins.codigoBarra);
                    comando.Parameters.AddWithValue("@nomProducto", ins.nomProducto);
                    comando.Parameters.AddWithValue("@fechaRegistro", ins.fechaRegistro);
                    comando.Parameters.AddWithValue("@idProveedores", ins.idProveedores);
                    comando.Parameters.AddWithValue("@Estatus", Convert.ToInt32(ins.Estatus));
                    comando.Connection = Conexion.conexion;
                    return comando.ExecuteNonQuery();
                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                if (ex.Number == 1062)
                {
                    Productos enf = new Productos();
                    enf = this.ObtenerUnProducto(ins.codigoBarra);
                    if (enf.Estatus.Equals(0))
                    {
                        MySqlCommand comando = new MySqlCommand(
                        "update proveedores set
nomProducto=@nomProducto,fechaRegistro=@fechaRegistro,idProveedores=@idProveedores,Estatus=1 " +
                        "where codigoBarra=@codigoBarra");
                        comando.Parameters.AddWithValue("@codigoBarra", ins.codigoBarra);
                        comando.Parameters.AddWithValue("@nomProducto", ins.nomProducto);
                        comando.Parameters.AddWithValue("@fechaRegistro", ins.fechaRegistro);
                        comando.Parameters.AddWithValue("@idProveedores", ins.idProveedores);
                        comando.Connection = Conexion.conexion;
                        return comando.ExecuteNonQuery();
                    }
                    else
                    {
                        return 0;
                    }
                }
                else
                {
                    throw new Exception("No se ha podido agregar el producto");
                }
            }
            finally
            {
                Conexion.desconectar();
            }

        }

        /// <summary>
        /// Método para actualizar un registro de la tabla de Productos
        /// </summary>
        /// <param name="ins">
        /// objeto de tipo Productos que recibirá todos los datos para la actualización
        /// del registro de la tabla Productos.
        /// </param>
        /// <returns>
        /// Regresa una variable de tipo Boolean con valor 1 0 true si el registro se pudo actualizar,
        /// y un valor de 0 o false si no se pudo actualizar el registro
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de que no se el registro no se pudo actualizar mandara un mensaje donde indica
        /// que no se pudo realizar el registro.
        /// </exception>
        public bool ModificarProducto(Productos ins)
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand(
                   "update productos set nomProducto=@nomProducto,idProveedores=@idProveedores " +
                   "where codigoBarra=@codigoBarra");
                    comando.Parameters.AddWithValue("@codigoBarra", ins.codigoBarra);
                    comando.Parameters.AddWithValue("@nomProducto", ins.nomProducto);
                    comando.Parameters.AddWithValue("@idProveedores", ins.idProveedores);
                    comando.Connection = Conexion.conexion;
```

```
            int filasEditadas = comando.ExecuteNonQuery();
            return (filasEditadas > 0);

        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        throw new Exception("No se pudo actualizar wl producto");
    }
    finally
    {
        Conexion.desconectar();
    }
}

/// <summary>
/// Método que elimina un registro de la tabla de Productos
/// </summary>
/// <param name="codigo">
/// parámetro que recibe un String con un Código proporcionado para buscar un registro
/// para su posterior eliminación.
/// </param>
/// <returns>
/// Regresa una variable de tipo Boolean con valor 1 0 true si el registro se pudo eliminar,
/// y un valor de 0 o false si no se pudo eliminar el registro
/// </returns>
/// <exception cref="Exception">
/// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
/// conectar al servidor.
/// En caso de que no el registro no se pudo eliminar mandara un mensaje donde indica
/// que no se pudo realizar el registro.
/// </exception>
public bool EliminarProducto(string codigo)
{
    try
    {
        if (Conexion.conectar())
        {
            MySqlCommand comando = new MySqlCommand(
                @"delete from productos where codigoBarra=@codigo");
            comando.Parameters.AddWithValue("@codigo", codigo);
            comando.Connection = Conexion.conexion;
            int filasBorradas = comando.ExecuteNonQuery();
            return (filasBorradas > 0);

        }
        else
        {
            throw new Exception("No se ha podido conectar con el servidor");
        }
    }
    catch (MySqlException ex)
    {
        if (ex.Number == 1451)
        {
            try
            {
                MySqlCommand comando = new MySqlCommand(
                @"update productos set Estatus=false where codigoBarra=@codigo");
                comando.Parameters.AddWithValue("@codigo", codigo);
                comando.Connection = Conexion.conexion;
                int filasBorradas = comando.ExecuteNonQuery();
                return (filasBorradas > 0);
            }
            catch
            {
                throw new Exception("Producto ya eliminado");
            }
        }
        else
        {
            throw new Exception("Error al intentar eliminar el producto");
        }

    }
    finally
    {
        Conexion.desconectar();
    }
}
/// <summary>
/// Método que Realiza consultas para obtener registros de la tabla Productos.
/// </summary>
/// <param name="text">
/// parámetro de entrada que recibe una variable de tipo String que contiene una cadena
/// de texto que se utilizara para buscar coincidencias con los registros de la tabla Productos.
/// </param>
/// <returns>
/// Regresa una lista de registros de la tabla Productos que coincidan con la cadena de texto que
```

```
        /// se proporciono.
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de que no se pudo realizar la conexión se mandara un mensaje donde indica que la
        /// conexión no se logró realizar.
        /// En caso de que no se encuentren datos en la tabla de Productos mandara un mensaje que
        /// indica que no se pudieron obtener dato de la tabla.
        /// </exception>
        public List<Productos> Buscar(string text)
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand(
                        @"select prod.*, prov.nomProveedor from productos prod join proveedores prov on prod.idProveedores =
prov.idProveedores
                              where
                              codigoBarra like  CONCAT('%', @ref, '%') ||
                              nomProducto like  CONCAT('%', @ref, '%') ||
                              nomProveedor like  CONCAT('%', @ref, '%');"
                    );
                    comando.Parameters.AddWithValue("@ref", text);
                    comando.Connection = Conexion.conexion;
                    MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                    DataTable resultado = new DataTable();
                    adapter.Fill(resultado);
                    List<Productos> lista = new List<Productos>();
                    Productos objProducto = null;

                    foreach (DataRow fila in resultado.Rows)
                    {
                        if (Convert.ToInt32(fila["Estatus"]).Equals(1))
                        {
                            objProducto = new Productos();
                            objProducto.codigoBarra = fila["codigoBarra"].ToString();
                            objProducto.nomProducto = fila["nomProducto"].ToString();
                            objProducto.fechaRegistro = fila["fechaRegistro"].ToString();
                            objProducto.idProveedores = fila["idProveedores"].ToString();
                            objProducto.nomProveedor = fila["nomProveedor"].ToString();
                            objProducto.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);

                            lista.Add(objProducto);
                        }
                    }

                    return lista;
                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                throw new Exception("No se pudo obtener la información de los productos");
            }
            finally
            {
                Conexion.desconectar();
            }
        }
    }
}
```

DAOProveedores

```
using Modelos;
using MySql.Data.MySqlClient;
using MySqlX.XDevAPI;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Datos
{
    /// <summary>
    /// Clase DAOProveedores que contiene los métodos para realizar consultas, inserciones, actualizaciones y eliminación
    /// de registros de la tabla Empleados de la base de datos del programa.
    /// </summary>
    public class DAOProveedores
    {
        /// <summary>
        /// Método que obtiene una lista con todos los registros de la tabla de Proveedores.
        /// </summary>
        /// <returns>
        /// Regresa una lista de Tipo Proveedores con todos los registros de la tabla Proveedores
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
        /// registros en la tabla
        /// </exception>
        public List<Proveedores> ObtenerProveedores()
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand("select * from proveedores");
                    comando.Connection = Conexion.conexion;
                    MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                    DataTable resultado = new DataTable();
                    adapter.Fill(resultado);
                    List<Proveedores> lista = new List<Proveedores>();
                    Proveedores objProveedores = null;
                    foreach (DataRow fila in resultado.Rows)
                    {
                        objProveedores = new Proveedores();
                        objProveedores.idProveedores = (fila["idProveedores"].ToString());
                        objProveedores.nomProveedores = fila["nomProveedor"].ToString();
                        objProveedores.RFC = fila["RFC"].ToString();
                        objProveedores.Direccion = fila["Direccion"].ToString();
                        objProveedores.Telefono = fila["Telefono"].ToString();
                        objProveedores.Correo = fila["Correo"].ToString();
                        objProveedores.fechaRegistro = fila["fechaRegistro"].ToString();
                        objProveedores.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);
                        lista.Add(objProveedores);
                    }

                    return lista;
                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                throw new Exception("No se pudo obtener la información de los proveedores");
            }
            finally
            {
                Conexion.desconectar();
            }
        }
        /// <summary>
        /// Metodo que obtiene una lista con todos los registros de la tabla de Proveedores que están activos.
        /// </summary>
        /// <returns>
        /// Regresa una lista de Tipo Proveedores con todos los registros de la tabla Proveedores que estén
        /// en estado activo
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
        /// registros en la tabla
        /// </exception>
        public List<Proveedores> ObtenerProveedoresActivos()
        {
            try
            {
```

```
            if (Conexion.conectar())
            {
                MySqlCommand comando = new MySqlCommand("select * from proveedores where Estatus=true");
                comando.Connection = Conexion.conexion;
                MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                DataTable resultado = new DataTable();
                adapter.Fill(resultado);
                List<Proveedores> lista = new List<Proveedores>();
                Proveedores objProveedores = null;
                foreach (DataRow fila in resultado.Rows)
                {
                    objProveedores = new Proveedores();
                    objProveedores.idProveedores = (fila["idProveedores"].ToString());
                    objProveedores.nomProveedores = fila["nomProveedor"].ToString();
                    objProveedores.RFC = fila["RFC"].ToString();
                    objProveedores.Direccion = fila["Direccion"].ToString();
                    objProveedores.Telefono = fila["Telefono"].ToString();
                    objProveedores.Correo = fila["Correo"].ToString();
                    objProveedores.fechaRegistro = fila["fechaRegistro"].ToString();
                    objProveedores.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);
                    lista.Add(objProveedores);
                }

                return lista;
            }
            else
            {
                throw new Exception("No se ha podido conectar con el servidor");
            }
        }
        catch (MySqlException ex)
        {
            throw new Exception("No se pudo obtener la información de los proveedores");
        }
        finally
        {
            Conexion.desconectar();
        }
    }
    /// <summary>
    /// Método que obtiene una lista con todos los registros de la tabla de Proveedores que están inactivos.
    /// </summary>
    /// <returns>
    /// Regresa una lista de Tipo Proveedores con todos los registros de la tabla Proveedores que estén
    /// en estado inactivo
    /// </returns>
    /// <exception cref="Exception">
    /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
    /// conectar al servidor.
    /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
    /// registros en la tabla
    /// </exception>
    public List<Proveedores> ObtenerProveedoresInactivos()
    {
        try
        {
            if (Conexion.conectar())
            {
                MySqlCommand comando = new MySqlCommand("select * from proveedores where Estatus=false");
                comando.Connection = Conexion.conexion;
                MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                DataTable resultado = new DataTable();
                adapter.Fill(resultado);
                List<Proveedores> lista = new List<Proveedores>();
                Proveedores objProveedores = null;
                foreach (DataRow fila in resultado.Rows)
                {
                    objProveedores = new Proveedores();
                    objProveedores.idProveedores = (fila["idProveedores"].ToString());
                    objProveedores.nomProveedores = fila["nomProveedor"].ToString();
                    objProveedores.RFC = fila["RFC"].ToString();
                    objProveedores.Direccion = fila["Direccion"].ToString();
                    objProveedores.Telefono = fila["Telefono"].ToString();
                    objProveedores.Correo = fila["Correo"].ToString();
                    objProveedores.fechaRegistro = fila["fechaRegistro"].ToString();
                    objProveedores.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);
                    lista.Add(objProveedores);
                }

                return lista;
            }
            else
            {
                throw new Exception("No se ha podido conectar con el servidor");
            }
        }
        catch (MySqlException ex)
        {
            throw new Exception("No se pudo obtener la información de los proveedores");
        }
        finally
        {
            Conexion.desconectar();
```

```
            }
        }

/// <summary>
        /// Método que obtiene un objeto con el registro de la tabla de Proveedores que coincidan con
        /// el ID proporcionado.
        /// </summary>
        /// <param name="id">
        /// parámetro que recibe un String con ID para buscar un registro en la tabla de Proveedores
        /// </param>
        /// <returns>
        /// Regresa un objeto de Tipo Proveedores con todos los registros de la tabla Proveedores que coincidan
        /// con el ID proporcionado.
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de no encontrar ningún registro regresa un mensaje que indica que no se encontraron
        /// registros en la tabla
        /// </exception>
        public Proveedores ObtenerUnProveedor(string id)
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand("select * from proveedores where idProveedores=@id");
                    comando.Parameters.AddWithValue("@id", id);
                    comando.Connection = Conexion.conexion;
                    MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                    DataTable resultado = new DataTable();
                    adapter.Fill(resultado);
                    Proveedores objProveedores = null;
                    if (resultado.Rows.Count > 0)
                    {
                        objProveedores = new Proveedores();
                        DataRow fila = resultado.Rows[0];
                        objProveedores = new Proveedores();

                        objProveedores.idProveedores = (fila["idProveedores"].ToString());
                        objProveedores.nomProveedores = (fila["nomProveedor"].ToString());
                        objProveedores.RFC = (fila["RFC"].ToString());
                        objProveedores.Direccion = (fila["Direccion"].ToString());
                        objProveedores.Telefono = (fila["Telefono"].ToString());
                        objProveedores.Correo = (fila["Correo"].ToString());
                        objProveedores.fechaRegistro = fila["fechaRegistro"].ToString();
                        objProveedores.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);
                    }

                    return objProveedores;
                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                throw new Exception("No se pudo obtener la informacion del proveedor");
            }
            finally
            {
                Conexion.desconectar();
            }
        }
        /// <summary>
        /// Método para realizar un nuevo registro en la tabla de Proveedores.
        /// </summary>
        /// <param name="ins">
        /// objeto de tipo Proveedores que recibirá todos los datos para el nuevo registro
        /// de la tabla Proveedores.
        /// </param>
        /// <returns>
        /// Regresa una int con un valor de 1 si el registro se pudo realizar y un o si no se realizó el registro
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de que no se el registro no se pudo realizar mandara un mensaje donde indica
        /// que no se pudo realizar el registro.
        /// </exception>
        public int AgregarProveedor(Proveedores ins)
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand(
                    "INSERT INTO proveedores VALUES(@idProveedores, @nomProveedor, @RFC, @Direccion, @Telefono, @Correo,
@fechaRegistro, @Estatus);");
                    comando.Parameters.AddWithValue("@idProveedores", ins.idProveedores);
                    comando.Parameters.AddWithValue("@nomProveedor", ins.nomProveedores);
                    comando.Parameters.AddWithValue("@RFC", ins.RFC);
```

```
                    comando.Parameters.AddWithValue("@Direccion", ins.Direccion);
                    comando.Parameters.AddWithValue("@Telefono", ins.Telefono);
                    comando.Parameters.AddWithValue("@Correo", ins.Correo);
                    comando.Parameters.AddWithValue("@fechaRegistro", ins.fechaRegistro);
                    comando.Parameters.AddWithValue("@Estatus", Convert.ToInt32(ins.Estatus));
                    comando.Connection = Conexion.conexion;
                    return comando.ExecuteNonQuery();
                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                if (ex.Number == 1062)
                {
                    Proveedores enf = new Proveedores();
                    enf = this.ObtenerUnProveedor(ins.idProveedores);
                    if (enf.Estatus.Equals(0))
                    {
                        MySqlCommand comando = new MySqlCommand(
                        "update proveedores set
nomProveedor=@nomProveedor,RFC=@RFC,Direccion=@Direccion,Telefono=@Telefono,Correo=@Correo,fechaRegistro=@fechaRegistro,Estat
us=1 " +
                        "where idProveedores=@idProveedores");
                        comando.Parameters.AddWithValue("@idProveedores", ins.idProveedores);
                        comando.Parameters.AddWithValue("@nomProveedor", ins.nomProveedores);
                        comando.Parameters.AddWithValue("@RFC", ins.RFC);
                        comando.Parameters.AddWithValue("@Direccion", ins.Direccion);
                        comando.Parameters.AddWithValue("@Telefono", ins.Telefono);
                        comando.Parameters.AddWithValue("@Correo", ins.Correo);
                        comando.Parameters.AddWithValue("@fechaRegistro", ins.fechaRegistro);
                        comando.Connection = Conexion.conexion;
                        return comando.ExecuteNonQuery();
                    }
                    else
                    {
                        return 0;
                    }
                }
                else
                {
                    throw new Exception("No se ha podido agregar al proveedor");
                }
            }
            finally
            {
                Conexion.desconectar();
            }

        }
        /// <summary>
        /// Método para actualizar un registro de la tabla de Proveedores
        /// </summary>
        /// <param name="ins">
        /// objeto de tipo Proveedores que recibirá todos los datos para la actualización
        /// del registro de la tabla Productos.
        /// </param>
        /// <returns>
        /// Regresa una variable de tipo Boolean con valor 1 0 true si el registro se pudo actualizar,
        /// y un valor de 0 o false si no se pudo actualizar el registro
        /// </returns>
        /// <exception cref="Exception">
        /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
        /// conectar al servidor.
        /// En caso de que no se el registro no se pudo actualizar mandara un mensaje donde indica
        /// que no se pudo realizar el registro.
        /// </exception>
        public bool ModificarProveedor(Proveedores ins)
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand(
                    "update proveedores set
nomProveedor=@nomProveedor,RFC=@RFC,Direccion=@Direccion,Telefono=@Telefono,Correo=@Correo " +
                    "where idProveedores=@idProveedores");
                    comando.Parameters.AddWithValue("@idProveedores", ins.idProveedores);
                    comando.Parameters.AddWithValue("@nomProveedor", ins.nomProveedores);
                    comando.Parameters.AddWithValue("@RFC", ins.RFC);
                    comando.Parameters.AddWithValue("@Direccion", ins.Direccion);
                    comando.Parameters.AddWithValue("@Telefono", ins.Telefono);
                    comando.Parameters.AddWithValue("@Correo", ins.Correo);
                    comando.Connection = Conexion.conexion;
                    int filasEditadas = comando.ExecuteNonQuery();
                    return (filasEditadas > 0);
                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
```

```
        }
        catch (MySqlException ex)
        {
            throw new Exception("No se pudo actualizar al proveedor");
        }
        finally
        {
            Conexion.desconectar();
        }
    }

    /// <summary>
    /// Método que elimina un registro de la tabla de Proveedores
    /// </summary>
    /// <param name="id">
    /// parámetro que recibe un String con un ID proporcionado para buscar un registro
    /// para su posterior eliminación.
    /// </param>
    /// <returns>
    /// Regresa una variable de tipo Boolean con valor 1 0 true si el registro se pudo eliminar,
    /// y un valor de 0 o false si no se pudo eliminar el registro
    /// </returns>
    /// <exception cref="Exception">
    /// En caso de no poder conectarse al sistema mostrara un mensaje donde indique que no se pudo
    /// conectar al servidor.
    /// En caso de que no el registro no se pudo eliminar mandara un mensaje donde indica
    /// que no se pudo realizar el registro.
    /// </exception>
    public bool EliminarProveedor(string id)
    {
        try
        {
            if (Conexion.conectar())
            {
                MySqlCommand comando = new MySqlCommand(
                    @"delete from proveedores where idProveedores=@Id");
                comando.Parameters.AddWithValue("@id", id);
                comando.Connection = Conexion.conexion;
                int filasBorradas = comando.ExecuteNonQuery();
                return (filasBorradas > 0);

            }
            else
            {
                throw new Exception("No se ha podido conectar con el servidor");
            }
        }
        catch (MySqlException ex)
        {
            if (ex.Number == 1451)
            {
                try
                {
                    MySqlCommand comando = new MySqlCommand(
                    @"update proveedores set Estatus=false where idProveedores=@Id");
                    comando.Parameters.AddWithValue("@id", id);
                    comando.Connection = Conexion.conexion;
                    int filasBorradas = comando.ExecuteNonQuery();
                    return (filasBorradas > 0);
                }
                catch
                {
                    throw new Exception("Proveedor ya eliminado");
                }
            }
            else
            {
                throw new Exception("Error al intentar eliminar al proveedor");
            }

        }
        finally
        {
            Conexion.desconectar();
        }
    }
    /// <summary>
    /// Método que Realiza consultas para obtener registros de la tabla Proveedores.
    /// </summary>
    /// <param name="text">
    /// parámetro de entrada que recibe una variable de tipo String que contiene una cadena
    /// de texto que se utilizara para buscar coincidencias con los registros de la tabla Proveedores.
    /// </param>
    /// <returns>
    /// Regresa una lista de registros de la tabla Proveedores que coincidan con la cadena de texto que
    /// se proporcionó.
    /// </returns>
    /// <exception cref="Exception">
    /// En caso de que no se pudo realizar la conexión se mandara un mensaje donde indica que la
    /// conexión no se logró realizar.
    /// En caso de que no se encuentren datos en la tabla de Proveedores mandara un mensaje que
    /// indica que no se pudieron obtener dato de la tabla.
    /// </exception>
```

```
        public List<Proveedores> Buscar(string text)
        {
            try
            {
                if (Conexion.conectar())
                {
                    MySqlCommand comando = new MySqlCommand(
                        @"select * from proveedores
                              where
                              idProveedores like  CONCAT('%', @ref, '%') ||
                              nomProveedor like  CONCAT('%', @ref, '%') ||
                              RFC like  CONCAT('%', @ref, '%') ||
                              Direccion like  CONCAT('%', @ref, '%') ||
                              Telefono like  CONCAT('%', @ref, '%') ||
                              Correo like  CONCAT('%', @ref, '%') ||
                              fechaRegistro like  CONCAT('%', @ref, '%');"
                    );
                    comando.Parameters.AddWithValue("@ref", text);
                    comando.Connection = Conexion.conexion;
                    MySqlDataAdapter adapter = new MySqlDataAdapter(comando);
                    DataTable resultado = new DataTable();
                    adapter.Fill(resultado);
                    List<Proveedores> lista = new List<Proveedores>();
                    Proveedores objProveedor = null;

                    foreach (DataRow fila in resultado.Rows)
                    {
                        if (Convert.ToInt32(fila["Estatus"]).Equals(1))
                        {
                            objProveedor = new Proveedores();
                            objProveedor.idProveedores = fila["idProveedores"].ToString();
                            objProveedor.nomProveedores = fila["nomProveedor"].ToString();
                            objProveedor.RFC = fila["RFC"].ToString();
                            objProveedor.Direccion = fila["Direccion"].ToString();
                            objProveedor.Telefono = fila["Telefono"].ToString();
                            objProveedor.Correo = fila["Correo"].ToString();
                            objProveedor.fechaRegistro = fila["fechaRegistro"].ToString();
                            objProveedor.Estatus = (Convert.ToInt32(fila["Estatus"]) == 1 ? true : false);

                            lista.Add(objProveedor);
                        }

                    }

                    return lista;
                }
                else
                {
                    throw new Exception("No se ha podido conectar con el servidor");
                }
            }
            catch (MySqlException ex)
            {
                throw new Exception("No se pudo obtener la información de los proveedores");
            }
            finally
            {
                Conexion.desconectar();
            }
        }
    }
}
```

Modelo Almacen

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Modelos
{
    /// <summary>
    /// Modelo de los datos de Almacén que sirve para guardar los datos que se guardaran en los registros
    /// de la base de datos
    /// </summary>
    public class Almacen
    {
        /// <summary>
        /// Parámetros o getters y setters que reciben los valores para guardar los registros
        /// de la tabla Almacén de la base de datos
        /// </summary>
        public string idAlmacen { get; set; }
        public int cantProducto { get; set; }
        public string codigoBarra { get; set; }
        public string idEmpleado { get; set; }
        public string Estatus { get; set; }

        /// <summary>
        /// Constructor de la clase vacío que no recibe ningún parámetro
        /// </summary>
        public Almacen() { }

        /// <summary>
        /// Constructor que recibirá parámetros para guardar algún registro.
        /// </summary>
        /// <param name="idAlmacen">
        /// parámetro que recibirá un String con id del almacén a almacenar en un registro
        /// </param>
        /// <param name="cantProducto">
        /// parámetro que recibirá un int con la cantidad de productos a almacén un registro
        /// </param>
        /// <param name="codigoBarra">
        /// parámetro que recibirá un string con el código de barra a almacén un registro
        /// </param>
        /// <param name="idEmpleado">
        /// parámetro que recibirá un string con el id de un empleado a almacén un registro
        /// </param>
        /// <param name="estatus">
        /// parámetro que recibirá un string que indica si el producto a almacenar esta activo o inactivo.
        /// </param>
        public Almacen(string idAlmacen, int cantProducto, string codigoBarra, string idEmpleado, string estatus)
        {
            this.idAlmacen = idAlmacen;
            this.cantProducto = cantProducto;
            this.codigoBarra = codigoBarra;
            this.idEmpleado = idEmpleado;
            Estatus = estatus;
        }
    }
}
```

Modelo Clientes

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Modelos
{
    /// <summary>
    /// Modelo de los datos de Cliente que sirve para guardar los datos que se guardaran en los registros
    /// de la base de datos de la tabla clientes
    /// </summary>
    public class Cliente
    {
        /// <summary>
        /// Parámetros o getters y setters que reciben los valores para guardar los registros
        /// de la tabla clientes de la base de datos
        /// </summary>
        public string idCliente { get; set; }
        public string RFC { get; set; }
        public string CURP { get; set; }
        public string nomCliente { get; set; }
        public string Direccion { get; set; }
        public string Telefono { get; set; }
        public string Correo { get; set; }
        public string Estatus { get; set; }

        /// <summary>
        /// Constructor del modelo que sirve para hacer referencia y crear instancias del modelo de clientes
        /// lo cual permite usar los getters y setters en otras clases
        /// </summary>
        public Cliente() { }

        /// <summary>
        /// Constructor que hace referencias al modelo de Cliente el cual se utilizara para almacenar los
        /// datos de un nuevo registro en la base de datos.
        /// </summary>
        /// <param name="idCliente">
        /// parámetro que recibirá un string con el id del cliente a almacenar en un registro.
        /// </param>
        /// <param name="rFC">
        /// parámetro que recibirá un string con el RFC del cliente a almacenar en un registro.
        /// </param>
        /// <param name="cURP">
        /// parámetro que recibirá un string con la CURP del cliente a almacenar en un registro.
        /// </param>
        /// <param name="nomCliente">
        /// parámetro que recibirá un string con el nombre del cliente a almacenar en un registro.
        /// </param>
        /// <param name="direccion">
        /// parámetro que recibirá un string con la dirección del cliente a almacenar en un registro.
        /// </param>
        /// <param name="telefono">
        /// parámetro que recibirá un string con el teléfono del cliente a almacenar en un registro.
        /// </param>
        /// <param name="correo">
        /// parámetro que recibirá un string con el correo del cliente a almacenar en un registro.
        /// </param>
        /// <param name="estatus">
        /// parámetro que recibirá un string con el estatus del cliente a almacenar en un registro
        /// el estado solo puede ser activo o inactivo.
        /// </param>
        public Cliente(string idCliente, string rFC, string cURP, string nomCliente, string direccion, string telefono,
string correo, string estatus)
        {
            this.idCliente = idCliente;
            RFC = rFC;
            CURP = cURP;
            this.nomCliente = nomCliente;
            Direccion = direccion;
            Telefono = telefono;
            Correo = correo;
            Estatus = estatus;
        }
    }
}
```

Modelo Contrato

```
using System;
using System.Collections.Generic;
using System.Diagnostics.Contracts;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Modelos
{
    /// <summary>
    /// Modelo de los datos de Contrato que sirve para guardar los datos que se guardaran en los registros
    /// de la base de datos de la tabla Contratos
    /// </summary>
    public class Contrato
    {
        /// <summary>
        /// Parámetros o getters y setters que reciben los valores para guardar los registros
        /// de la tabla Contratos de la base de datos
        /// </summary>
        public string idContrato { get; set; }
        public double precioServicio { get; set; }
        public string inicioContrato { get; set; }
        public string finContrato { get; set; }
        public string idCliente { get; set; }
        public String Estatus { get; set; }

        /// <summary>
        /// Constructor que hace referencias al modelo de Contrato el cual se utilizara para almacenar los
        /// datos de un nuevo registro en la base de datos.
        /// </summary>
        /// <param name="idContrato">
        /// Parametro que recivira un string con el id del contrato a almacenar en un registro.
        /// </param>
        /// <param name="precioServicio">
        /// Parametro que recivira un double con el precio del servicio del
        /// contrato a almacenar en un registro.
        /// </param>
        /// <param name="inicioContrato">
        /// parámetro que recibirá un string con la fecha de inicio
        /// del contrato a almacenar en un registro.
        /// </param>
        /// <param name="finContrato">
        /// parámetro que recibirá un string con la fecha de finalización del contrato
        /// contrato a almacenar en un registro.
        /// </param>
        /// <param name="idCliente">
        /// parámetro que recibirá un string con el id del cliente a almacenar en un registro.
        /// </param>
        /// <param name="estatus">
        /// parámetro que recibirá un string con el estatus del Contrato a almacenar en un registro
        /// el estado solo puede ser activo o inactivo.
        /// </param>
        public Contrato(string idContrato, double precioServicio, string inicioContrato, string finContrato, string
idCliente, string estatus)
        {
            this.idContrato = idContrato;
            this.precioServicio = precioServicio;
            this.inicioContrato = inicioContrato;
            this.finContrato = finContrato;
            this.idCliente = idCliente;
            Estatus = estatus;
        }

        /// <summary>
        /// Constructor del modelo que sirve para hacer referencia y crear instancias del modelo de Contrato
        /// lo cual permite usar los getters y setters en otras clases
        /// </summary>
        public Contrato() { }
    }
}
```

Modelo Empleados

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Runtime.InteropServices;
using System.Security.Policy;
using System.Text;
using System.Threading.Tasks;

namespace Modelos
{
    /// <summary>
    /// Modelo de los datos de Empleados que sirve para guardar los datos que se guardaran en los registros
    /// de la base de datos de la tabla Empleados
    /// </summary>
    public class Empleados
    {
        /// <summary>
        /// Parámetros o getters y setters que reciben los valores para guardar los registros
        /// de la tabla Empleados de la base de datos
        /// </summary>
        public string Idempleado { get; set; }
        public string Nombrecompleto { get; set; }
        public string RFC { get; set; }
        public string CURP { get; set; }
        public int Edad { get; set; }
        public string Direccion { get; set; }
        public string Telefono { get; set; }
        public string Correo { get; set; }
        public string fechaContratacion { get; set; }
        public string Rol { get; set; }
        public string Password { get; set; }
        public bool Estatus { get; set; }

        /// <summary>
        /// Constructor del modelo que sirve para hacer referencia y crear instancias del modelo de Empleados
        /// lo cual permite usar los getters y setters en otras clases
        /// </summary>
        public Empleados(){}

        /// <summary>
        /// Constructor que hace referencias al modelo de Empleados el cual se utilizara para almacenar los
        /// datos de un nuevo registro en la base de datos.
        /// </summary>
        /// <param name="IdEmpleado">
        /// parámetro que recibirá un string con el id del Empleado a almacenar en un registro.
        /// </param>
        /// <param name="nombreCompleto">
        /// parámetro que recibirá un string con el nombre del Empleados a almacenar en un registro.
        /// </param>
        /// <param name="Rfc">
        /// parámetro que recibirá un string con el RFC del Empleado a almacenar en un registro.
        /// </param>
        /// <param name="Curp">
        /// parámetro que recibirá un string con la CURP del Empleado a almacenar en un registro.
        /// </param>
        /// <param name="edad">
        /// parámetro que recibirá un int con la edad del Empleado a almacenar en un registro.
        /// </param>
        /// <param name="direccion">
        /// parámetro que recibirá un string con la dirección del Empleado a almacenar en un registro.
        /// </param>
        /// <param name="telefono">
        /// parámetro que recibirá un string con el teléfono del Empleado a almacenar en un registro.
        /// </param>
        /// <param name="correo">
        /// parámetro que recibirá un string con el correo del Empleado a almacenar en un registro.
        /// </param>
        /// <param name="fechacontra">
        /// parámetro que recibirá un string con la fecha de contratación
        /// del Empleado a almacenar en un registro.
        /// </param>
        /// <param name="rol">
        /// parámetro que recibirá un string con el Rol o puesto del Empleado a almacenar en un registro.
        /// </param>
        /// <param name="password">
        /// parámetro que recibirá un string con la contraseña del Empleado a almacenar en un registro.
        /// </param>
        /// <param name="estatus">
        /// parámetro que recibirá un Boolean con el estatus del Empleado a almacenar en un registro
        /// el estado solo puede ser activo o inactivo.
        /// </param>
        public Empleados(string IdEmpleado, string nombreCompleto, string Rfc, string Curp, int edad,
            string direccion, string telefono,string correo, string fechacontra, string rol, string password, bool estatus)
        {
            Idempleado = IdEmpleado;
            Nombrecompleto = nombreCompleto;
            RFC = Rfc;
            CURP = Curp;
            Edad = edad;
```

```
            Direccion = direccion;
            Telefono = telefono;
            Correo = correo;
            fechaContratacion = fechacontra;
            Rol = rol;
            Password = password;
            Estatus = estatus;
        }
        /// <summary>
        /// Constructor que recibe dos parámetros los cuales sirven para hacer un login.
        /// </summary>
        /// <param name="idempleado">
        /// parámetro que recibirá un string con el id del Empleado a almacenar en un registro.
        /// </param>
        /// <param name="nombreCompleto">
        /// parámetro que recibirá un string con el nombre del Empleados a almacenar en un registro.
        /// </param>
        public Empleados(string idempleado, string nombreCompleto)
        {
            this.Idempleado = idempleado;
            Nombrecompleto = nombreCompleto;
        }
    }
}
```

Modelo Instalaciones

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Modelos
{
    /// <summary>
    /// Modelo de los datos de Instalaciones que sirve para guardar los datos que se guardaran en los registros
    /// de la base de datos de la tabla Instalaciones
    /// </summary>
    public class Instalaciones
    {
        /// <summary>
        /// Parametros o getters y setters que resiven los valores para guardar los registros
        /// de la tabla Instalaciones de la base de datos
        /// </summary>
        public string IdInstalacion { get; set; }
        public string fechaInstalacin { get; set; }
        public string idEmpelado { get; set; }

        public string idContratos { get; set; }
        public bool Estatus { get; set; }

        /// <summary>
        /// Constructor del modelo que sirve para hacer referencia y crear instancias del modelo de Empleados
        /// lo cual permite usar los getters y setters en otras clases
        /// </summary>
        public Instalaciones() { }

        /// <summary>
        /// Constructor que hace referencias al modelo de Instalaciones el cual se utilizara para almacenar los
        /// datos de un registro en la base de datos.
        /// </summary>
        /// <param name="idInstalacion">
        /// Parametro que recibirá un string con el id la Instalación a almacenar en un registro.
        /// </param>
        /// <param name="fechaInstalacin">
        /// Parametro que recibirá un string con la fecha de la Instalación
        /// a almacenar en un registro.
        /// </param>
        /// <param name="idEmpelado">
        /// Parametro que recibirá un string con el id del Empleado a almacenar en un registro.
        /// </param>
        /// <param name="idContratos">
        /// Parametro que recibirá un string con el id del Contrato a almacenar en un registro.
        /// </param>
        /// <param name="estatus">
        /// Parametro que recibirá un Boolean con el estatus de la Instalación a almacenar en un registro
        /// el estado solo puede ser activo o inactivo.
        /// </param>
        public Instalaciones(string idInstalacion, string fechaInstalacin, string idEmpelado, string idContratos, bool
estatus)
        {
            IdInstalacion = idInstalacion;
            this.fechaInstalacin = fechaInstalacin;
            this.idEmpelado = idEmpelado;
            this.idContratos = idContratos;
            Estatus = estatus;
        }
    }
}
```

Modelo Productos

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Modelos
{
    /// <summary>
    /// Modelo de los datos de Productos que sirve para almacenar los datos que se guardaran en los registros
    /// de la base de datos de la tabla Productos
    /// </summary>
    public class Productos
    {
        /// <summary>
        /// Parámetros o getters y setters que reciben los valores para guardar los registros
        /// de la tabla Productos de la base de datos
        /// </summary>
        public string codigoBarra { get; set; }
        public string nomProducto { get; set; }
        public string fechaRegistro { get; set; }
        public string idProveedores { get; set; }
        public string nomProveedor { get; set; }
        public bool Estatus { get; set; }

        /// <summary>
        /// Constructor del modelo que sirve para hacer referencia y crear instancias del modelo de Empleados
        /// lo cual permite usar los getters y setters en otras clases
        /// </summary>
        public Productos() { }

        /// <summary>
        /// Constructor que hace referencias al modelo de Productos el cual se utilizara para almacenar los
        /// datos de un registro en la base de datos.
        /// </summary>
        /// <param name="codigoBarra">
        /// Parametro que recibirá un string con el código de barras del producto a almacenar en un registro.
        /// </param>
        /// <param name="nomProducto">
        /// Parametro que recibirá un string con el nombre del Producto a almacenar en un registro.
        /// </param>
        /// <param name="fechaRegistro">
        /// Parametro que recibirá un string con la fecha de registro de un producto a almacenar
        /// </param>
        /// <param name="idProveedores">
        /// Parametro que recibirá un string con el id del proveedor a almacenar en un registro.
        /// </param>
        /// <param name="nomProveedor">
        /// Parametro que recibirá un string con el nombre del Proveedor a almacenar en un registro.
        /// </param>
        /// <param name="estatus">
        /// Parametro que recibirá un Boolean con el estatus del producto a almacenar en un registro
        /// el estado solo puede ser activo o inactivo.
        /// </param>
        public Productos(string codigoBarra, string nomProducto, string fechaRegistro, string idProveedores, string
nomProveedor, bool estatus)
        {
            this.codigoBarra = codigoBarra;
            this.nomProducto = nomProducto;
            this.fechaRegistro = fechaRegistro;
            this.idProveedores = idProveedores;
            this.nomProveedor = nomProveedor;
            Estatus = estatus;
        }
    }
}
```

Modelo Proveedores

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Modelos
{
    /// <summary>
    /// Modelo de los datos de Proveedores que sirve para almacenar los datos que se guardaran en los registros
    /// de la base de datos de la tabla Empleados
    /// </summary>
    public class Proveedores
    {
        /// <summary>
        /// Parametros o getters y setters que reciben los valores para guardar los registros
        /// de la tabla Proveedores de la base de datos
        /// </summary>
        public string idProveedores { get; set; }
        public string nomProveedores { get; set; }
        public string RFC { get; set; }
        public string Direccion { get; set; }
        public string Telefono { get; set; }
        public string Correo { get; set; }
        public string fechaRegistro { get; set; }
        public bool Estatus { get; set; }

        /// <summary>
        /// Constructor del modelo que sirve para hacer referencia y crear instancias del modelo de Proveedores
        /// lo cual permite usar los getters y setters en otras clases
        /// </summary>
        public Proveedores() { }

        /// <summary>
        /// Constructor que hace referencias al modelo de Proveedores el cual se utilizara para almacenar los
        /// datos de un nuevo registro en la base de datos.
        /// </summary>
        /// <param name="idProveedores">
        /// Parametro que recibirá un string con el id del Proveedor a almacenar en un registro.
        /// </param>
        /// <param name="nomProveedores">
        /// Parametro que recibirá un string con el nombre del Proveedor a almacenar en un registro.
        /// </param>
        /// <param name="RFC">
        /// Parametro que recibirá un string con el RFC del Proveedor a almacenar en un registro.
        /// </param>
        /// <param name="Direccion">
        /// Parametro que recibirá un string con la dirección del Proveedor a almacenar en un registro.
        /// </param>
        /// <param name="Telefono">
        /// Parametro que recibirá un string con el teléfono del Proveedor a almacenar en un registro.
        /// </param>
        /// <param name="Correo">
        /// Parametro que recibirá un string con el correo del Proveedor a almacenar en un registro.
        /// </param>
        /// <param name="fechaRegistro">
        /// Parametro que recibirá un string con la fecha de Registro
        /// del Proveedor a almacenar en un registro.
        /// </param>
        /// <param name="Estatus">
        /// Parametro que recibirá un Boolean con el estatus del Proveedor a almacenar en un registro
        /// el estado solo puede ser activo o inactivo.
        /// </param>
        public Proveedores(string idProveedores, string nomProveedores, string RFC, string Direccion, string Telefono, string
Correo, string fechaRegistro, bool Estatus)
        {
            this.idProveedores = idProveedores;
            this.nomProveedores = nomProveedores;
            this.RFC = RFC;
            this.Direccion = Direccion;
            this.Telefono = Telefono;
            this.Correo = Correo;
            this.fechaRegistro = fechaRegistro;
            this.Estatus = Estatus;
        }
    }
}
```

Formulario de Login

```
using Datos;
using Modelos;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SGMDT_Wifimex
{
    /// <summary>
    /// Formulario con el Longin del sistema
    /// </summary>
    public partial class frmLogin : Form
    {
        /// <summary>
        /// Constructor que inicializa el formulario cargando todos los controles en el formulario
        /// </summary>
        public frmLogin()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Evento del botón Ingresar, al dar click intentara ingresar al sistema.
        /// en este evento se hace una referencia al DAO de Empleados y al modelo de Empleados
        /// para hacer una consulta y validación con las cajas de texto del formulario
        /// en caso de que la información que este en las cajas de texto coincida con un
        /// registro de la tabla de Empleados dará acceso al sistema, en caso de lo contrario
        /// denegara el acceso al sistema.
        /// </summary>
        private void btnIngresar_Click(object sender, EventArgs e)
        {
            DAOEmpleados dao = new DAOEmpleados();
            Empleados empleados = dao.IniciarSeccion(txtUsuario.Text, txtPassword.Text);
            if (empleados != null)
            {
                MessageBox.Show("Bienvenido usuario " + empleados.Idempleado);
                frmMenuPrincipal ini = new frmMenuPrincipal(empleados.Idempleado, empleados.Nombrecompleto);
                ini.Show();
                this.Hide();
            }
            else
            {
                MessageBox.Show("Usuario y/o Contraseña incorrectos");
            }
        }
    }
}
```

Formulario menu principal

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SGMDT_Wifimex
{
    /// <summary>
    /// Formulario de menú principal del programa el cual tiene varios botones
    /// que direccionaran a los diferentes menús de registros
    /// </summary>
    public partial class frmMenuPrincipal : Form
    {
        private String Idempelado,nombreCompleto;

        /// <summary>
        /// Evento del botón Empleados, al dar click al botón
        /// abrirá y direccionara al formulario principal de
        /// Empleados.
        /// </summary>
        private void btnEmpleados_Click(object sender, EventArgs e)
        {
            frmEmpleados frm = new frmEmpleados();
            frm.Show();
        }

        /// <summary>
        /// Evento del botón Instalaciones, al dar click al botón
        /// abrirá y direccionará al formulario principal de
        /// Instalaciones.
        /// </summary>
        private void btnInstalaciones_Click(object sender, EventArgs e)
        {
            frmInstalaciones frm = new frmInstalaciones();
            frm.Show();
        }

        /// <summary>
        /// Evento de cerrar el formulario, al cerrar el formulario
        /// se aplicación o programa se cerrara.
        /// </summary>
        private void frmMenuPrincipal_FormClosed(object sender, FormClosedEventArgs e)
        {
            Application.Exit();
        }

        /// <summary>
        /// Evento del boton Clientes, al dar click al boton
        /// abrira y direccionara al formulario principal de
        /// Clientes.
        /// </summary>
        private void btnClientes_Click(object sender, EventArgs e)
        {
            frmClientes frm = new frmClientes();
            frm.Show();
        }

        /// <summary>
        /// Evento del boton Contratos, al dar click al boton
        /// abrira y direccionara al formulario principal de
        /// Contratos.
        /// </summary>
        private void btnContratos_Click(object sender, EventArgs e)
        {
            frmContratos frm = new frmContratos();
            frm.Show();
        }

        /// <summary>
        /// Evento del boton Alamacen, al dar click al boton
        /// abrira y direccionara al formulario principal de
        /// Alamcen.
        /// </summary>
        private void btnAlmacen_Click(object sender, EventArgs e)
        {
            frmAlmacen frm= new frmAlmacen();
            frm.Show();
        }

        /// <summary>
        /// Evento del boton Productos, al dar click al boton
        /// abrira y direccionara al formulario principal de
        /// Productos.
        /// </summary>
        private void btnProductos_Click(object sender, EventArgs e)
```

```
        {
            frmProductos frm = new frmProductos();
            frm.Show();
        }

        /// <summary>
        /// Evento del boton Proveedores, al dar click al boton
        /// abrira y direccionara al formulario principal de
        /// Proveedores.
        /// </summary>
        private void btnProveedores_Click(object sender, EventArgs e)
        {
            frmProveedores frm = new frmProveedores();
            frm.Show();
        }

        /// <summary>
        /// Constructor del formulario que recibe dos parámetros.
        /// </summary>
        /// <param name="idempleado">
        /// Parametro donde se alamcena el ID del empleado que esta usando el sistema.
        /// </param>
        /// <param name="nombrecompleto">
        /// Parametro donde se almacena el ID del empleado que esta usando el sistema.
        /// </param>
        public frmMenuPrincipal(string idempleado, string nombrecompleto)
        {
            InitializeComponent();
                Idempelado = idempleado;
                nombreCompleto = nombrecompleto;
        }
    }
}
```

Formulario Menu Principal Almacen

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Datos;

namespace SGMDT_Wifimex
{
    /// <summary>
    /// Formulario de menú principal de Almacén, donde se darán las opciones de agregar, actualizar,
    /// Buscar y eliminar registros de la tabla de almacén.
    /// </summary>
    public partial class frmAlmacen : Form
    {
        /// <summary>
        /// Constructor del formulario donde se cargarán los controles que estarán visibles en
        /// el formulario, también cargara los datos de la tabla de almacén en una tabla para su
        /// visualización
        /// </summary>
        public frmAlmacen()
        {
            InitializeComponent();
            cargarTabla();
        }

        /// <summary>
        /// Mentodo que sirve para cargar los datos de la tabla almacen en una tabla del sistema
        /// haciendo una instancia del DAO de almacen al metodo obtener todos los registros.
        /// </summary>
        public void cargarTabla()
        {
            dgvAlmacen.DataSource= new DAOAlmacen().ObtenerTodos();
            dgvAlmacen.Columns["idAlmacen"].HeaderText = "Clave Almacen";
            dgvAlmacen.Columns["cantProducto"].HeaderText = "Cantidad en almacen";
            dgvAlmacen.Columns["codigoBarra"].HeaderText = "Clave Recurso";
            dgvAlmacen.Columns["idEmpleado"].HeaderText = "Clave empleado Encargado";
        }

        /// <summary>
        /// evento del boton Añadir, al dar click en este boton abrira la ventana de
        /// agrega/modificar un registro de almacén. haciendo una instancia del
        /// formulario de agregar/modificar almacén, cundo se finalice
        /// el trámite de añadir un nuevo registro se volverá a cargar la tabla
        /// con los nuevos registros
        /// </summary>
        private void btnAdd_Click(object sender, EventArgs e)
        {
            frmAddModyAlmacen frm= new  frmAddModyAlmacen("", 1);
            frm.ShowDialog();
            if (frm.Agregado>0)
            {
                cargarTabla();
            }
        }

        /// <summary>
        /// evento del boton Modificar, al seleccionar un registro
        /// de la tabla y dar click en este boton abrira la ventana de
        /// agrega/modificar un registro de almacén. haciendo una instancia del
        /// formulario de agregar/modificar almacén, cundo se finalice
        /// el trámite de modificar un registro se volverá a cargar la tabla
        /// con los registros actualizados.
        /// </summary>
        private void btnModify_Click(object sender, EventArgs e)
        {
            if (dgvAlmacen.SelectedRows.Count == 1)
            {
                String idEmpleado = dgvAlmacen.SelectedRows[0].Cells["idAlmacen"].Value.ToString();
                frmAddModyAlmacen frm = new frmAddModyAlmacen(idEmpleado, 2);
                frm.ShowDialog();
                if (frm.modificado)
                {
                    cargarTabla();
                }
            }
            else
            {
                MessageBox.Show("Se debe seleccionar un elemento de la tabla editar", "", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
            }
        }

        /// <summary>
        /// Evento del boton eliminar, al dar click a eliminar se eliminará el registro seleccionado
        /// de la tabla. Al seleccionar un registro de la tabla y dar click a eliminar emergerá un mensaje
```

```
        /// donde se pregunta si se quiere eliminar el registro, al seleccionar si el registro se eliminara
        /// de la base de datos y cargara la tabla con los registros actualizados, en caso de seleccionar
        /// no la acción se abortara y no habrá  ningún cambio.
        /// </summary>
        private void btnDelete_Click(object sender, EventArgs e)
        {
            if (dgvAlmacen.SelectedRows.Count == 1)
            {
                DialogResult dialog = MessageBox.Show("Estas a punto de eliminar al Almacen con clave  \n\r" +
                                                      dgvAlmacen.SelectedRows[0].Cells["idAlmacen"].Value.ToString() + ",
¿deseas contunuar?", "", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
                if (DialogResult.Yes == dialog)
                {
                    bool resp = new DAOAlmacen().eliminar(dgvAlmacen.SelectedRows[0].Cells["idAlmacen"].Value.ToString());
                    if (resp)
                    {
                        MessageBox.Show("Almacen eliminado con exito", "", MessageBoxButtons.OK, MessageBoxIcon.Question);
                        cargarTabla();
                    }
                    else
                    {
                        MessageBox.Show("El Almacen no se a podido eliminar", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                    }
                }
            }
            else
            {
                MessageBox.Show("Se debe seleccionar un elemento de la tabla para continuar", "", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
            }
        }
    }
}
```

Formulario de Añadir y Modificar alamcen

```
using Datos;
using Modelos;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SGMDT_Wifimex
{
    /// <summary>
    /// Formulario para la agregar y modificar los registros de la tabla de Almacen
    /// </summary>
    public partial class frmAddModyAlmacen : Form
    {
        /// <summary>
        /// Variables que sirven para validar la opción que se mostrara en el formulario
        /// ya se Agregar o modificar.
        /// </summary>
        public Boolean modificado;
        public int Agregado;
        private int opcion;

        /// <summary>
        /// Metodo que sirve para validar que las cajas de texto del formulario tengan
        /// el formato correcto que se solicita
        /// </summary>
        /// <returns>
        /// regresa un true si los formatos de las cajas de texto son validos en caso de
        /// lo contrario regresa un false.
        /// </returns>
        public bool Verificar()
        {
            if (!Regex.IsMatch(txtClave.Text, "^[A-Za-z]{4}[0-9]{6}$"))
            {
                errorProvider1.SetError(txtClave, "El formato debe contener 4 letras y 6 numeros en este orden");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }
            return true;
        }


        /// <summary>
        /// Costrcutor del formulario donde cargan todos los controles que
        /// estan dentro del formulario.
        /// </summary>
        /// <param name="ID">
        /// Parametro que recibe el constructor en caso de que se de la opcion de modificar
        /// un registro para buscar el registro a modificar
        /// </param>
        /// <param name="op">
        /// Parametro que indica cuales controles tiene que cargar el formulario
        /// este indica si el formulario tiene que ponerse en modo de agregar un
        /// nuevo registro o en modo de modificar un registro ya existente.
        /// </param>
        public frmAddModyAlmacen(string ID, int op)
        {
            InitializeComponent();
            cbxEmpleado.DataSource = new DAOEmpleados().ObtenerEmpleados();
            cbxEmpleado.DisplayMember = "nomEmpleados";
            cbxEmpleado.ValueMember = "idEmpleado";
            cbxEmpleado.SelectedIndex= 0;


            cbxBarra.DataSource = new DAOProductos().ObtenerProductos();
            cbxBarra.DisplayMember = "nomProducto";
            cbxBarra.ValueMember = "codigoBarra";
            cbxBarra.SelectedIndex = 0;

            opcion = op;
            switch (op)
            {
                case 1:
                    this.Text = "Agregar";
                    gbxEstatus.Enabled = false;
                    break;
                case 2:
                    this.Text = "Actualizar";
                    txtClave.Enabled = false;
                    Almacen cli = new DAOAlmacen().ObtenerUno(ID);
```

```
                    txtClave.Text = cli.idAlmacen.ToString();
                    nudCant.Text = cli.cantProducto.ToString();
                    cbxBarra.SelectedValue = cli.codigoBarra.ToString();
                    cbxEmpleado.SelectedValue = cli.idEmpleado.ToString();
                    if (cli.Estatus.ToString() == "Activo")
                    { rbActivo.Checked = true; }
                    else if (cli.Estatus.ToString() == "Inactivo")
                    { rbInactivo.Checked = true; }

                    break;
                default:
                    break;
            }
        }

        /// <summary>
        /// Evento click de boton, al dar click mandara y guardara el nuevo registro o modificara
        /// un registro ya existente.
        /// En caso de que la opcion seleccionada sea agregar se hace una referencia al modelo de almacen
        /// así como al DAO almacén para poder utilizar el metodo de agregar un nuevo registro.
        /// Si la opcion seleccionada es modificar también hará referencias a las mismas clases ya
        /// mencionadas, con el cambio de que ahora llamara al metodo de actualizar un registro.
        /// Al finalizar lanzar un mensaje de registro exitoso o fallo de registro, al cerrar
        /// el mensaje regresara al menú principal de almacen.
        /// </summary>
        private void btnAcep_Click(object sender, EventArgs e)
        {
            try
            {
                if (opcion==1)
                {
                    Almacen almacen = new Almacen();
                    almacen.idAlmacen = txtClave.Text;
                    almacen.cantProducto = int.Parse(nudCant.Value.ToString());
                    almacen.codigoBarra = cbxBarra.SelectedValue.ToString();
                    almacen.idEmpleado = cbxEmpleado.SelectedValue.ToString();
                    if (Verificar())
                    {
                        Agregado = new DAOAlmacen().agregar(almacen);
                        if (Agregado > 0)
                        {
                            MessageBox.Show("Almacen agregado con exito", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                            this.Close();
                        }
                        else
                        {
                            MessageBox.Show("El almacen no se a podido agregar", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        }
                    }
                }
                else
                {
                    Almacen almacen = new Almacen();
                    almacen.idAlmacen = txtClave.Text;
                    almacen.cantProducto = int.Parse(nudCant.Value.ToString());
                    almacen.codigoBarra = cbxBarra.SelectedValue.ToString();
                    almacen.idEmpleado = cbxEmpleado.SelectedValue.ToString();
                    almacen.Estatus = rbActivo.Checked == true ? "Activo" : "Inactivo";
                if (Verificar())
                    {
                        modificado = new DAOAlmacen().editar(almacen);
                        if (modificado)
                        {
                            MessageBox.Show("Almacen modificado con exito", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                            this.Close();
                        }
                        else
                        {
                            MessageBox.Show("El almacen no se a podido modificar", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        }
                    }
                }
        }
            catch (Exception ex)
            {

                MessageBox.Show(ex.Message, "Mensaje Error", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
}
        /// <summary>
        /// Evento de boton cancelar, al dar click al boton cierra la ventana y regresa al menú principal
        /// de Almacen.
        /// </summary>
        private void btnCancel_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Formulario de menu principal de Clientes

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Datos;

namespace SGMDT_Wifimex
{
    /// <summary>
    /// Formulario de menú principal de Clientes, donde se darán las opciones de agregar, actualizar,
    /// Buscar y eliminar registros de la tabla de Clientes.
    /// </summary>
    public partial class frmClientes : Form
    {
        /// <summary>
        /// Constructor del formulario donde se cargarán los controles que estarán visibles en
        /// el formulario, también cargara los datos de la tabla de almacen en una tabla para su
        /// visualización
        /// </summary>
        public frmClientes()
        {
            InitializeComponent();
            cargarTabla();

        }

        /// <summary>
        /// Mentado que sirve para cargar los datos de la tabla Clientes en una tabla del sistema
        /// haciendo una instancia del DAO de Clientes al metodo obtener todos los registros.
        /// </summary>
        public void cargarTabla()
        {
            DAOCliente dat = new DAOCliente();
            dgvClientes.DataSource = new DAOCliente().ObtenerTodos();
            dgvClientes.Columns["nomCliente"].HeaderText = "Nombre del cliente";
            dgvClientes.Columns["idCliente"].HeaderText = "Clave del Cliente";
        }

        /// <summary>
        /// evento del boton Añadir, al dar click en este boton abrira la ventana de
        /// agrega/modificar un registro de Clientes. haciendo una instancia del
        /// formulario de agregar/modificar Clientes, cundo se finalice
        /// el trámite de añadir un nuevo registro se volverá a cargar la tabla
        /// con los nuevos registros
        /// </summary>
        private void btnAdd_Click(object sender, EventArgs e)
        {
            frmAddModyCliente frm = new frmAddModyCliente("", 1);
            frm.ShowDialog();
            if (frm.Agregado>0)
            {
                cargarTabla();
            }
        }

        /// <summary>
        /// evento del boton Modificar, al seleccionar un registro
        /// de la tabla y dar click en este boton abrira la ventana de
        /// agrega/modificar un registro de Clientes. haciendo una instancia del
        /// formulario de agregar/modificar Clientes, cundo se finalice
        /// el trámite de modificar un registro se volverá a cargar la tabla
        /// con los registros actualizados.
        /// </summary>
        private void btnModify_Click(object sender, EventArgs e)
        {
            if (dgvClientes.SelectedRows.Count == 1)
            {
                String idEmpleado = dgvClientes.SelectedRows[0].Cells["idCliente"].Value.ToString();
                frmAddModyCliente frm = new frmAddModyCliente(idEmpleado, 2);
                frm.ShowDialog();
                if (frm.modificado)
                {
                    cargarTabla();
                }
            }
            else
            {
                MessageBox.Show("Se debe seleccionar un elemento de la tabla editar", "", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
            }
        }

        /// <summary>
        /// Evento del boton eliminar, al dar click a eliminar se eliminará el registro seleccionado
        /// de la tabla. Al seleccionar un registro de la tabla y dar click a eliminar emergerá un mensaje
```

```
        /// donde se pregunta si se quiere eliminar el registro, al seleccionar si el registro se eliminara
        /// de la base de datos y cargara la tabla con los registros actualizados, en caso de seleccionar
        /// no la acción se abortara y no habrá ningún cambio.
        /// </summary>
        private void btnDelete_Click(object sender, EventArgs e)
        {
            if (dgvClientes.SelectedRows.Count == 1)
            {
                DialogResult dialog = MessageBox.Show("Estas a punto de eliminar al cliente \n\r" +
                                                      dgvClientes.SelectedRows[0].Cells["nomCliente"].Value.ToString() + ",
¿deseas contunuar?", "", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
                if (DialogResult.Yes==dialog)
                {
                    bool resp = new DAOCliente().eliminar(dgvClientes.SelectedRows[0].Cells["idCliente"].Value.ToString());
                    if (resp) {
                        MessageBox.Show("Cliente eliminado con exito", "", MessageBoxButtons.OK, MessageBoxIcon.Question);
                        cargarTabla();
                    }
                    else
                    {
                        MessageBox.Show("el cliente no se a podido eliminar", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                    }
                }
            }
            else
            {
                MessageBox.Show("Se debe seleccionar un elemento de la tabla para continuar", "", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
            }
        }

        /// <summary>
        /// Evento de caja de texto, cuando se escribe en la caja de texto,
        /// al escribir en la caja de texto llamara al metodo de buscar
        /// del DAO de clientes, si encuentra coincidencias con algún registro
        /// de la base de datos de la tabla Clientes, cargara los registros que coincidan
        /// en la tabla de sistema.
        /// </summary>
        private void txtBus_TextChanged(object sender, EventArgs e)
        {
            if (txtBus.Text.Length >= 4)
            {
                DAOCliente dat = new DAOCliente();
                dgvClientes.DataSource = new DAOCliente().Buscar(txtBus.Text);
                dgvClientes.Columns["nomCliente"].HeaderText = "Nombre del cliente";
                dgvClientes.Columns["idCliente"].HeaderText = "Clave del Cliente";
            }
            else
            {
                cargarTabla();
            }
        }
    }
}
```

Formulario de Añadir y Modificar Cliente

```
using Datos;
using Modelos;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Linq.Expressions;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SGMDT_Wifimex
{
    /// <summary>
    /// Formulario que agrega y modifica los registros de la tabla clientes
    /// </summary>
    public partial class frmAddModyCliente : Form
    {
        /// <summary>
        /// Variables que sirven para validar la opcion que se mostrara en el formulario
        /// ya se Agregar o modificar.
        /// </summary>
        public Boolean modificado;
        public int Agregado;

        private bool resp=false;

        /// <summary>
        /// Metodo que sirve para validar que las cajas de texto del formulario tengan
        /// el formato correcto que se solicita
        /// </summary>
        /// <returns>
        /// regresa un true si los formatos de las cajas de texto son válidos en caso de
        /// lo contrario regresa un false.
        /// </returns>
        public bool Verificar()
        {
            if (!Regex.IsMatch(txtId.Text, "^[A-Za-z]{4}[0-9]{6}$"))
            {
                errorProvider1.SetError(txtId, "El formato debe contener 4 letras y 6 numeros en este orden");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }
            if (!Regex.IsMatch(txtRfc.Text, @"^[A-Z]{4}\d{6}([A-Z0-9]{2}[A-Z0-9]{1})?$"))
            {
                errorProvider1.SetError(txtRfc, "Verifique que su RFC este correcto");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }
            if (!Regex.IsMatch(txtCurp.Text, @"^[A-Z]{4}\d{6}[HM][A-Z]{5}[A-Z0-9]{2}$"))
            {
                errorProvider1.SetError(txtCurp, "Verifique que su CURP cumpla con el formato requerido");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }
            if (!Regex.IsMatch(txtNom.Text, @"^[A-Za-z][A-Za-z .]{1,}[A-Za-z]$"))
            {
                errorProvider1.SetError(txtNom, "El nombre no puede contener números o caracteres especiales");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }
            if (!Regex.IsMatch(txtDir.Text, @"^[A-Za-z][A-Za-z .]{1,}\s+#\d+(-\d+)?$"))
            {
                errorProvider1.SetError(txtDir, "Siga al formato adecuado , ejemplo : 'Calle #123', en caso de ser
departamento agregue '-Número interior'");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }
            if (!Regex.IsMatch(txtTel.Text, @"^[0-9]{10}?$"))
            {
                errorProvider1.SetError(txtTel, "Verifique el número contenga 10 dígitos");
                return false;
```

```
        }
        else
        {
            errorProvider1.Clear();
        }
        if (!Regex.IsMatch(txtCorr.Text, @"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"))
        {
            errorProvider1.SetError(txtCorr, "Verifique en correo este correctamente escrito y sin espacios");
            return false;
        }
        else
        {
            errorProvider1.Clear();
        }


        return true;
    }

    private int opcion;

    /// <summary>
    /// Constructor del formulario donde cargan todos los controles que
    /// están dentro del formulario.
    /// </summary>
    /// <param name="ID">
    /// Parametro que recibe el constructor en caso de que se de la opcion de modificar
    /// un registro para buscar el registro a modificar
    /// </param>
    /// <param name="Opcion">
    /// Parametro que indica cuales controles tiene que cargar el formulario
    /// este indica si el formulario tiene que ponerse en modo de agregar un
    /// nuevo registro o en modo de modificar un registro ya existente.
    /// </param>
    public frmAddModyCliente(string ID, int Opcion)
    {
        InitializeComponent();
        opcion= Opcion;
        switch (Opcion)
        {
            case 1:
                this.Text = "Agregar";
                gbxEstatus.Enabled = false;
                break;
            case 2:
                this.Text = "Actualizar";
                txtId.Enabled = false;
                Cliente cli = new DAOCliente().ObtenerUno(ID);

                txtId.Text = cli.idCliente.ToString();
                txtRfc.Text = cli.RFC.ToString();
                txtCurp.Text = cli.CURP.ToString();
                txtNom.Text = cli.nomCliente.ToString();
                txtDir.Text = cli.Direccion.ToString();
                txtTel.Text = cli.Telefono.ToString();
                txtCorr.Text = cli.Correo.ToString();
                if(cli.Estatus.ToString() == "Activo")
                { rbActivo.Checked = true; }
                else if (cli.Estatus.ToString() == "Inactivo")
                { rbInactivo.Checked = true; }

                break;
            default:
                break;
        }

    }

    /// <summary>
    /// Evento click de boton, al dar click mandara y guardara el nuevo registro o modificara
    /// un registro ya existente.
    /// En caso de que la opcion seleccionada sea agregar se hace una referencia al modelo de Clientes
    /// así como al DAO Clientes para poder utilizar el metodo de agregar un nuevo registro.
    /// Si la opcion seleccionada es modificar también hará referencias a las mismas clases ya
    /// mencionadas, con el cambio de que ahora llamara al metodo de actualizar un registro.
    /// Al finalizar lanzar un mensaje de registro exitoso o fallo de registro, al cerrar
    /// el mensaje regresara al menú principal de Clientes.
    /// </summary>
    private void button2_Click(object sender, EventArgs e)
    {
        try
        {
            if (opcion == 1)
            {
                Cliente cli = new Cliente();
                cli.idCliente = txtId.Text;
                cli.RFC = txtRfc.Text;
                cli.CURP = txtCurp.Text;
                cli.nomCliente = txtNom.Text;
                cli.Direccion = txtDir.Text;
                cli.Telefono = txtTel.Text;
                cli.Correo = txtCorr.Text;
                if (Verificar())
```

```
                    {
                        Agregado = new DAOCliente().agregar(cli);
                        if (Agregado > 0)
                        {
                            MessageBox.Show("Cliente Agregado con exito", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                            this.Close();
                        }
                        else
                        {
                            MessageBox.Show("El cliente no se a podido agregar", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        }
                    }
                }
                else
                {
                    Cliente cli = new Cliente();
                    cli.idCliente = txtId.Text;
                    cli.RFC = txtRfc.Text;
                    cli.CURP = txtCurp.Text;
                    cli.nomCliente = txtNom.Text;
                    cli.Direccion = txtDir.Text;
                    cli.Telefono = txtTel.Text;
                    cli.Correo = txtCorr.Text;
                    cli.Estatus = rbActivo.Checked == true ? "Activo" : "Inactivo";
                    if (Verificar())
                    {
                        modificado = new DAOCliente().editar(cli);
                        if (modificado)
                        {
                            MessageBox.Show("Cliente modificado con exito", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                            this.Close();
                        }
                        else
                        {
                            MessageBox.Show("El cliente no se a podido modificar", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        }
                    }
                }
            }
            catch (Exception ex)
            {

                MessageBox.Show(ex.Message, "Mensaje", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }

        /// <summary>
        /// Evento de boton cancelar, al dar click al boton cierra la ventana y regresa al menú principal
        /// de Clientes.
        /// </summary>
        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Formulario de Menu principal Contratos

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Datos;

namespace SGMDT_Wifimex
{
    /// <summary>
    /// Formulario de menú principal de Clientes, donde se darán las opciones de agregar, actualizar,
    /// Buscar y eliminar registros de la tabla de Clientes.
    /// </summary>
    public partial class frmContratos : Form
    {
        /// <summary>
        /// Constructor del formulario donde se cargarán los controles que estarán visibles en
        /// el formulario, también cargara los datos de la tabla de Contratos en una tabla para su
        /// visualización
        /// </summary>
        public frmContratos()
        {
            InitializeComponent();
            CargarTabla();
        }

        /// <summary>
        /// Mentado que sirve para cargar los datos de la tabla Contratos en una tabla del sistema
        /// haciendo una instancia del DAO de Contratos al metodo obtener todos los registros.
        /// </summary>
        public void CargarTabla()
        {
            dgvContratos.DataSource= new DAOContratos().ObtenerTodos();
            dgvContratos.Columns["idContrato"].HeaderText = "Clave del contrato";
            dgvContratos.Columns["precioServicio"].HeaderText = "Costo por el servicio";
            dgvContratos.Columns["inicioContrato"].HeaderText = "Inicio del contrato";
            dgvContratos.Columns["finContrato"].HeaderText = "Fin del contrato";
            dgvContratos.Columns["idCliente"].HeaderText = "Clave del cliente";
        }

        /// <summary>
        /// evento del boton Añadir, al dar click en este boton abrira la ventana de
        /// agrega/modificar un registro de Contratos. haciendo una instancia del
        /// formulario de agregar/modificar Contratos, cundo se finalice
        /// el trámite de añadir un nuevo registro se volverá a cargar la tabla
        /// con los nuevos registros
        /// </summary>
        private void btnAdd_Click(object sender, EventArgs e)
        {
            frmAddModyContratos frm = new frmAddModyContratos("", 1);
            frm.ShowDialog();
            if (frm.Agregado>0)
            {
                CargarTabla();
            }
        }

        /// <summary>
        /// evento del boton Modificar, al seleccionar un registro
        /// de la tabla y dar click en este boton abrira la ventana de
        /// agrega/modificar un registro de Contratos. haciendo una instancia del
        /// formulario de agregar/modificar Contratos, cundo se finalice
        /// el trámite de modificar un registro se volverá a cargar la tabla
        /// con los registros actualizados.
        /// </summary>
        private void btnModify_Click(object sender, EventArgs e)
        {
            if (dgvContratos.SelectedRows.Count == 1)
            {
                String idEmpleado = dgvContratos.SelectedRows[0].Cells["idContrato"].Value.ToString();
                frmAddModyContratos frm = new frmAddModyContratos(idEmpleado, 2);
                frm.ShowDialog();
                if (frm.modificado)
                {
                    CargarTabla();
                }
            }
            else
            {
                MessageBox.Show("Se debe seleccionar un elemento de la tabla editar", "", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
            }
        }

        /// <summary>
        /// Evento del boton eliminar, al dar click a eliminar se eliminara el registro seleccionado
```

```
        /// de la tabla. Al seleccionar un registro de la tabla y dar click a eliminar emergerá un mensaje
        /// donde se pregunta si se quiere eliminar el registro, al seleccionar si el registro se eliminara
        /// de la base de datos y cargara la tabla con los registros actualizados, en caso de seleccionar
        /// no la acción se abortara y no habrá ningún cambio.
        /// </summary>
        private void btnDelete_Click(object sender, EventArgs e)
        {
            if (dgvContratos.SelectedRows.Count == 1)
            {
                DialogResult dialog = MessageBox.Show("Estas a punto de eliminar el contrato \n\r" +
                                                      dgvContratos.SelectedRows[0].Cells["idContrato"].Value.ToString() + ",
¿deseas contunuar?", "", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
                if (DialogResult.Yes == dialog)
                {
                    bool resp = new
DAOContratos().eliminar(dgvContratos.SelectedRows[0].Cells["idContrato"].Value.ToString());
                    if (resp)
                    {
                        MessageBox.Show("Contrato eliminado con exito", "", MessageBoxButtons.OK, MessageBoxIcon.Question);
                        CargarTabla();
                    }
                    else
                    {
                        MessageBox.Show("El Contrato no se a podido eliminar", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                    }
                }
            }
            else
            {
                MessageBox.Show("Se debe seleccionar un elemento de la tabla para continuar", "", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
            }
        }

        /// <summary>
        /// Evento de caja de texto, cuando se escribe en la caja de texto,
        /// al escribir en la caja de texto llamara al metodo de buscar
        /// del DAO de Contratos, si encuentra coincidencias con algún registro
        /// de la base de datos de la tabla Contratos, cargara los registros que coincidan
        /// en la tabla de sistema.
        /// </summary>
        private void txtBus_TextChanged(object sender, EventArgs e)
        {
            if (txtBus.Text.Length >= 4)
            {
                dgvContratos.DataSource = new DAOContratos().Buscar(txtBus.Text);
                dgvContratos.Columns["idContrato"].HeaderText = "Clave del contrato";
                dgvContratos.Columns["precioServicio"].HeaderText = "Costo por el servicio";
                dgvContratos.Columns["inicioContrato"].HeaderText = "Inicio del contrato";
                dgvContratos.Columns["finContrato"].HeaderText = "Fin del contrato";
                dgvContratos.Columns["idCliente"].HeaderText = "Clave del cliente";
            }
            else
            {
                CargarTabla();
            }
        }
    }
}
```

Formulario de Añadir y Modificar Contratos

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;
using Datos;
using Modelos;

namespace SGMDT_Wifimex
{
    /// <summary>
    /// Formulario que agrega y modifica los registros de la tabla Contratos
    /// </summary>
    public partial class frmAddModyContratos : Form
    {
        /// <summary>
        /// Variables que sirven para validar la opcion que se mostrara en el formulario
        /// ya se Agregar o modificar.
        /// </summary>
        public Boolean modificado;
        public int Agregado;

        /// <summary>
        /// Metodo que sirve para validar que las cajas de texto del formulario tengan
        /// el formato correcto que se solicita
        /// </summary>
        /// <returns>
        /// regresa un true si los formatos de las cajas de texto son validos en caso de
        /// lo contrario regresa un false.
        /// </returns>
        public bool Verificar()
        {

            if (!Regex.IsMatch(txtIdCo.Text, "^[A-Za-z]{4}[0-9]{6}$"))
            {
                errorProvider1.SetError(txtIdCo, "El formato debe contener 4 letras y 6 numeros en este orden");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }
            if (!Regex.IsMatch(txtPrecio.Text, "^[0-9]{1,3}([.][0-9]{1,2})?$"))
            {
                errorProvider1.SetError(txtPrecio, "El valor debe ser menor o igual a 999.99");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            };

            if (dtpFin.Value<dtpIni.Value)
            {
                errorProvider1.SetError(dtpFin, "La fecha de fin debe de ser despues de la de inicio");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }


            return true;

        }

        private int opcion;

        /// <summary>
        /// Constructor del formulario donde cargan todos los controles que
        /// están dentro del formulario.
        /// </summary>
        /// <param name="ID">
        /// Parametro que recibe el constructor en caso de que se dé la opcion de modificar
        /// un registro para buscar el registro a modificar
        /// </param>
        /// <param name="Opcion">
        /// Parametro que indica cuales controles tiene que cargar el formulario
        /// este indica si el formulario tiene que ponerse en modo de agregar un
        /// nuevo registro o en modo de modificar un registro ya existente.
        /// </param>
        public frmAddModyContratos(string ID, int Opcion)
        {
            InitializeComponent();
```

```
            cbxCliente.DataSource = new DAOCliente().ObtenerTodos();
            cbxCliente.DisplayMember= "nomCliente";
            cbxCliente.ValueMember= "idCLiente";

            opcion= Opcion;
            switch (Opcion)
            {
                case 1:
                    this.Text = "Agregar";
                    lblTitulo.Text = "Crear Contrato";
                    gbxEstatus.Enabled = false;
                    cbxCliente.SelectedIndex = 0;
                    break;
                case 2:
                    this.Text = "Actualizar";
                    lblTitulo.Text = "Actualizar Contrato";
                    txtIdCo.Enabled = false;
                    Contrato con = new DAOContratos().ObtenerUno(ID);

                    txtIdCo.Text = con.idContrato.ToString();
                    txtPrecio.Text = con.precioServicio.ToString();
                    dtpIni.Text = con.inicioContrato.ToString();
                    dtpFin.Text = con.finContrato.ToString();
                    cbxCliente.SelectedValue = con.idCliente.ToString();
                    if (con.Estatus.ToString() == "Activo")
                    { rbActivo.Checked = true; }
                    else if (con.Estatus.ToString() == "Inactivo")
                    { rbInactivo.Checked = true; }

                    break;
                default:
                    break;
            }
        }

        /// <summary>
        /// Evento click de boton, al dar click mandara y guardara el nuevo registro o modificara
        /// un registro ya existente.
        /// En caso de que la opcion seleccionada sea agregar se hace una referencia al modelo de Contratos
        /// así como al DAO Contratos para poder utilizar el metodo de agregar un nuevo registro.
        /// Si la opcion seleccionada es modificar también hará referencias a las mismas clases ya
        /// mencionadas, con el cambio de que ahora llamara al metodo de actualizar un registro.
        /// Al finalizar lanzar un mensaje de registro exitoso o fallo de registro, al cerrar
        /// el mensaje regresara al menú principal de Contratos.
        /// </summary>
        private void button2_Click(object sender, EventArgs e)
        {
            try
            {
                if (opcion == 1)
                {
                    Contrato con = new Contrato();
                    con.idContrato = txtIdCo.Text;
                    con.precioServicio = double.Parse(txtPrecio.Text);
                    con.inicioContrato = dtpIni.Text;
                    con.idCliente = cbxCliente.SelectedValue.ToString();
                    con.finContrato = dtpFin.Text;
                    if (Verificar())
                    {
                        Agregado = new DAOContratos().agregar(con);
                        if (Agregado > 0)
                        {
                            MessageBox.Show("Contrato Agregado con exito", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                            this.Close();
                        }
                        else
                        {
                            MessageBox.Show("El contrato no se a podido agregar", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        }
                    }
                }
                else
                {
                    Contrato con = new Contrato();
                    con.idContrato = txtIdCo.Text;
                    con.precioServicio = double.Parse(txtPrecio.Text);
                    con.inicioContrato = dtpIni.Text;
                    con.idCliente = cbxCliente.SelectedValue.ToString();
                    con.finContrato = dtpFin.Text;
                    con.Estatus = rbActivo.Checked == true ? "Activo" : "Inactivo";
                    if (Verificar())
                    {
                        modificado = new DAOContratos().editar(con);
                        if (modificado)
                        {
                            MessageBox.Show("Contrato modificado con exito", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                            this.Close();
                        }
                        else
                        {
```

```
                            MessageBox.Show("El contrato no se a podido modificar", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        }
                    }
                }
            }
            catch (Exception ex)
            {

                MessageBox.Show(ex.Message, "Mensaje", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }

        /// <summary>
        /// Evento de boton cancelar, al dar click al boton cierra la ventana y regresa al menú principal
        /// de Contratos.
        /// </summary>
        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Formulario de menú principal de Empelados

```
using Datos;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SGMDT_Wifimex
{
    /// <summary>
    /// Formulario de menu principal de Empleados, donde se darán las opciones de agregar, actualizar,
    /// Buscar y eliminar registros de la tabla de Empleados.
    /// </summary>
    public partial class frmEmpleados : Form
    {
        /// <summary>
        /// Constructor del formulario donde se cargarán los controles que estarán visibles en
        /// el formulario, también cargara los datos de la tabla de Empleados en una tabla para su
        /// visualización
        /// </summary>
        public frmEmpleados()
        {
            InitializeComponent();
            CargarTabla();
        }

        /// <summary>
        /// Mentodo que sirve para cargar los datos de la tabla Empleados en una tabla del sistema
        /// haciendo una instancia del DAO de Empleados al metodo obtener todos los registros
        /// que estén en estado activos.
        /// </summary>
        public void CargarTablaAc()
        {
            dgvEmpleados.DataSource = new DAOEmpleados().ObtenerEmpleadosActivos();
            dgvEmpleados.Columns["idEmpleado"].Visible = false;
            dgvEmpleados.Columns["Nombrecompleto"].HeaderText = "Nombre Empleado";
            dgvEmpleados.Columns["RFC"].HeaderText = "RFC";
            dgvEmpleados.Columns["CURP"].HeaderText = "CURP";
            dgvEmpleados.Columns["Edad"].Visible = false;
            dgvEmpleados.Columns["Direccion"].Visible = false;
            dgvEmpleados.Columns["Telefono"].HeaderText = "Telefono";
            dgvEmpleados.Columns["Correo"].HeaderText = "Correo";
            dgvEmpleados.Columns["fechaContratacion"].HeaderText = "Fecha de Contratacion";
            dgvEmpleados.Columns["Rol"].HeaderText = "Rol";
            dgvEmpleados.Columns["Password"].Visible = false;
            dgvEmpleados.Columns["Estatus"].Visible = false;
        }

        /// <summary>
        /// Mentodo que sirve para cargar los datos de la tabla Empleados en una tabla del sistema
        /// haciendo una instancia del DAO de Empleados al metodo obtener todos los registros.
        /// </summary>
        public void CargarTabla()
        {
            dgvEmpleados.DataSource = new DAOEmpleados().ObtenerEmpleados();
            dgvEmpleados.Columns["idEmpleado"].Visible = false;
            dgvEmpleados.Columns["Nombrecompleto"].HeaderText = "Nombre Empleado";
            dgvEmpleados.Columns["RFC"].HeaderText = "RFC";
            dgvEmpleados.Columns["CURP"].HeaderText = "CURP";
            dgvEmpleados.Columns["Edad"].Visible = false;
            dgvEmpleados.Columns["Direccion"].Visible = false;
            dgvEmpleados.Columns["Telefono"].HeaderText = "Telefono";
            dgvEmpleados.Columns["Correo"].HeaderText = "Correo";
            dgvEmpleados.Columns["fechaContratacion"].HeaderText = "Fecha de Contratacion";
            dgvEmpleados.Columns["Rol"].HeaderText = "Rol";
            dgvEmpleados.Columns["Password"].Visible = false;
            dgvEmpleados.Columns["Estatus"].Visible = false;
        }
        /// <summary>
        /// Mentodo que sirve para cargar los datos de la tabla Empleados en una tabla del sistema
        /// haciendo una instancia del DAO de Empleados al metodo obtener todos los registros
        /// en estado inactivos.
        /// </summary>
        public void CargarTablaIn()
        {
            dgvEmpleados.DataSource = new DAOEmpleados().ObtenerEmpleadosInactivos();
            dgvEmpleados.Columns["idEmpleado"].Visible = false;
            dgvEmpleados.Columns["Nombrecompleto"].HeaderText = "Nombre Empleado";
            dgvEmpleados.Columns["RFC"].HeaderText = "RFC";
            dgvEmpleados.Columns["CURP"].HeaderText = "CURP";
            dgvEmpleados.Columns["Edad"].Visible = false;
            dgvEmpleados.Columns["Direccion"].Visible = false;
            dgvEmpleados.Columns["Telefono"].HeaderText = "Telefono";
            dgvEmpleados.Columns["Correo"].HeaderText = "Correo";
            dgvEmpleados.Columns["fechaContratacion"].HeaderText = "Fecha de Contratacion";
            dgvEmpleados.Columns["Rol"].HeaderText = "Rol";
```

```
    dgvEmpleados.Columns["Password"].Visible = false;
    dgvEmpleados.Columns["Estatus"].Visible = false;
}

/// <summary>
/// Mentodo que sirve para cargar los datos de la tabla Empleados en una tabla del sistema
/// haciendo una instancia del DAO de Empleados al metodo obtener todos los registros.
/// </summary>
public void CargarTablaEm()
{
    dgvEmpleados.DataSource = new DAOEmpleados().ObtenerEmpleado(txtBuscar.Text);
    dgvEmpleados.Columns["idEmpleado"].Visible = false;
    dgvEmpleados.Columns["Nombrecompleto"].HeaderText = "Nombre Empleado";
    dgvEmpleados.Columns["RFC"].HeaderText = "RFC";
    dgvEmpleados.Columns["CURP"].HeaderText = "CURP";
    dgvEmpleados.Columns["Edad"].Visible = false;
    dgvEmpleados.Columns["Direccion"].Visible = false;
    dgvEmpleados.Columns["Telefono"].HeaderText = "Telefono";
    dgvEmpleados.Columns["Correo"].HeaderText = "Correo";
    dgvEmpleados.Columns["fechaContratacion"].HeaderText = "Fecha de Contratacion";
    dgvEmpleados.Columns["Rol"].HeaderText = "Rol";
    dgvEmpleados.Columns["Password"].Visible = false;
    dgvEmpleados.Columns["Estatus"].Visible = false;
}

/// <summary>
/// evento del boton Añadir, al dar click en este boton abrira la ventana de
/// agrega/modificar un registro de Empleados. haciendo una instancia del
/// formulario de agregar/modificar Empleados, cundo se finalice
/// el trámite de añadir un nuevo registro se volverá a cargar la tabla
/// con los nuevos registros
/// </summary>
private void btnAgregar_Click(object sender, EventArgs e)
{
    frmAgregarModificar frm = new frmAgregarModificar(1, "0");
    frm.ShowDialog();
    if (frm.Guardado == 0)
    {
        CargarTabla();
    }
}

/// <summary>
/// evento del boton Modificar, al seleccionar un registro
/// de la tabla y dar click en este boton abrira la ventana de
/// agrega/modificar un registro de Empleados. haciendo una instancia del
/// formulario de agregar/modificar Empleados, cundo se finalice
/// el trámite de modificar un registro se volverá a cargar la tabla
/// con los registros actualizados.
/// </summary>
private void btnActualizar_Click(object sender, EventArgs e)
{
    if (dgvEmpleados.SelectedRows.Count > 0)
    {
        string idCategoria = dgvEmpleados.SelectedRows[0].Cells["idEmpleado"].Value.ToString();
        frmAgregarModificar frm = new frmAgregarModificar(2, idCategoria);
        frm.ShowDialog();
        if (frm.Modificado)
        {
            CargarTabla();
        }
    }
}

/// <summary>
/// Evento de boton mostrar todo, cuando se da click a este boton
/// mostrara todos los registros de la tabla empleados sin importar
/// si están activos o inactivos
/// </summary>
private void btnTodos_Click(object sender, EventArgs e)
{
    CargarTabla();
}

/// <summary>
/// Evento de boton buscar, cuando se da click a este boton
/// mostrara todos los registros que coincida con lo que está en la
/// caja de texto de la tabla empleados sin importar
/// si están activos o inactivos
/// </summary>
private void btnBusccar_Click(object sender, EventArgs e)
{
    CargarTablaEm();
}

/// <summary>
/// Evento del boton eliminar, al dar click a eliminar se eliminará el registro seleccionado
/// de la tabla. Al seleccionar un registro de la tabla y dar click a eliminar emergerá un mensaje
/// donde se pregunta si se quiere eliminar el registro, al seleccionar si el registro se eliminara
/// de la base de datos y cargara la tabla con los registros actualizados, en caso de seleccionar
/// no la acción se abortara y no habrá ningún cambio.
/// </summary>
private void btnEliminar_Click(object sender, EventArgs e)
```

```
        {
            if (dgvEmpleados.SelectedRows.Count > 0)
            {
                String idEmleado = dgvEmpleados.SelectedRows[0].Cells["idEmpleado"].Value.ToString();
                DialogResult resp = MessageBox.Show("Estas a punto de eliminar a " +
                                    (new DAOEmpleados().ObtenerUnEmpleado(idEmleado).Nombrecompleto) +
                                    " de las listas, ¿Deseas continuar?", "",
                                    MessageBoxButtons.YesNo, MessageBoxIcon.Question);
                if (resp == DialogResult.Yes)
                {
                    if (new DAOEmpleados().EliminarEmpleado(idEmleado))
                    {
                        MessageBox.Show("Se a eliminado el empleado seleccionada", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        CargarTabla();
                    }
                }
            }
        }

        /// <summary>
        /// Evento de boton mostrar todo, cuando se da click a este boton
        /// mostrara todos los registros de la tabla empleados inactivos
        /// </summary>
        private void btnInactivos_Click(object sender, EventArgs e)
        {
            CargarTablaAc();
        }

        /// <summary>
        /// Evento de boton mostrar todo, cuando se da click a este boton
        /// mostrara todos los registros de la tabla empleados inactivos
        /// </summary>
        private void btnActivos_Click(object sender, EventArgs e)
        {
            CargarTablaIn();
        }
    }
}
```

Formulario de Añandir y Modificar Empleados

```
using Datos;
using Modelos;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SGMDT_Wifimex
{
    /// <summary>
    /// Formulario que agrega y modifica los registros de la tabla Empleados
    /// </summary>
    public partial class frmAgregarModificar : Form
    {
        /// <summary>
        /// Variables que sirven para validar la opcion que se mostrara en el formulario
        /// ya se Agregar o modificar.
        /// </summary>
        private int OP;
        private string ID;
        public int Guardado { get; set; }
        public bool Modificado { get; set; }

        /// <summary>
        /// Costrcutor del formulario donde cargan todos los controles que
        /// estan dentro del formulario.
        /// </summary>
        /// <param name="id">
        /// Parametro que recibe el constructor en caso de que se dé la opcion de modificar
        /// un registro para buscar el registro a modificar
        /// </param>
        /// <param name="op">
        /// Parametro que indica cuales controles tiene que cargar el formulario
        /// este indica si el formulario tiene que ponerse en modo de agregar un
        /// nuevo registro o en modo de modificar un registro ya existente.
        /// </param>
        public frmAgregarModificar(int op, string id)
        {
            InitializeComponent();
            ID = id;
            OP = op;

            if (op==1)
            {
                this.Text = "Agregar";
                cbxRol.SelectedIndex = 0;
            }
            // Aquie se carga los datos para cuando se actualizan
            else if (op == 2)
            {
                this.Text = "Modificar";
                txtContrasena.Visible = false;
                label11.Visible = false;
                Empleados emp = new DAOEmpleados().ObtenerUnEmpleado(id);
                txtNumEmpleado.Text = emp.Idempleado;
                txtNombre.Text = emp.Nombrecompleto;
                txtCorreo.Text = emp.Correo;
                txtRFC.Text = emp.RFC;
                txtCURP.Text = emp.CURP;
                nudEdad.Text = (emp.Edad).ToString();
                txtDireccion.Text = emp.Direccion;
                txtTelefono.Text = emp.Telefono;
                cbxRol.SelectedItem = emp.Rol;
                dtpContrato.Enabled = false;
                txtNumEmpleado.Enabled = false;
            }
        }

        /// <summary>
        /// Metodo que sirve para validar que las cajas de texto del formulario tengan
        /// el formato correcto que se solicita
        /// </summary>
        /// <returns>
        /// regresa un true si los formatos de las cajas de texto son validos en caso de
        /// lo contrario regresa un false.
        /// </returns>
        public bool Verificar()
        {
            if (!Regex.IsMatch(txtNumEmpleado.Text, "^[A-Za-z]{4}[0-9]{6}$"))
            {
                erpVerificar.SetError(txtNumEmpleado, "El formato debe contener 4 letras y 6 numeros en este orden");
                Console.WriteLine("Falla");
                return false;
            }
```

```
            else
            {
                erpVerificar.Clear();
            }
            if (!Regex.IsMatch(txtRFC.Text, @"^[A-Z]{4}\d{6}([A-Z0-9]{2}[A-Z0-9]{1})?$"))
            {
                erpVerificar.SetError(txtRFC, "Verifique que su RFC este correcto");
                return false;
            }
            else
            {
                erpVerificar.Clear();
            }
            if (!Regex.IsMatch(txtCURP.Text, @"^[A-Z]{4}\d{6}[HM][A-Z]{5}[A-Z0-9]{2}$"))
            {
                erpVerificar.SetError(txtCURP, "Verifique que su CURP cumpla con el formato requerido");
                return false;
            }
            else
            {
                erpVerificar.Clear();
            }
            if (!Regex.IsMatch(txtNombre.Text, @"^[A-Za-z][A-Za-z .]{1,}[A-Za-z]$"))
            {
                erpVerificar.SetError(txtNombre, "El nombre no puede contener números o caracteres especiales");
                return false;
            }
            else
            {
                erpVerificar.Clear();
            }
            if (!Regex.IsMatch(txtDireccion.Text, @"^[A-Za-z][A-Za-z .]{1,}\s+#\d+(-\d+)?$"))
            {
                erpVerificar.SetError(txtDireccion, "Siga al formato adecuado , ejemplo : 'Calle #123', en caso de ser
departamento agregue '-Número interior'");
                return false;
            }
            else
            {
                erpVerificar.Clear();
            }
            if (!Regex.IsMatch(txtTelefono.Text, @"^[0-9]{10}?$"))
            {
                erpVerificar.SetError(txtTelefono, "Verifique el número contenga 10 dígitos");
                return false;
            }
            else
            {
                erpVerificar.Clear();
            }
            if (!Regex.IsMatch(txtCorreo.Text, @"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"))
            {
                erpVerificar.SetError(txtCorreo, "Verifique en correo este correctamente escrito y sin espacios");
                return false;
            }
            else
            {
                erpVerificar.Clear();
            }


            return true;
        }

        /// <summary>
        /// Evento click de boton, al dar click mandara y guardara el nuevo registro o modificara
        /// un registro ya existente.
        /// En caso de que la opcion seleccionada sea agregar se hace una referencia al modelo de Empleados
        /// así como al DAO Empleados para poder utilizar el metodo de agregar un nuevo registro.
        /// Si la opcion seleccionada es modificar también hará referencias a las mismas clases ya
        /// mencionadas, con el cambio de que ahora llamara al metodo de actualizar un registro.
        /// Al finalizar lanzar un mensaje de registro exitoso o fallo de registro, al cerrar
        /// el mensaje regresara al menu principal de Empleados.
        /// </summary>
        private void btnGuardar_Click(object sender, EventArgs e)
        {

                if (OP == 1)
                {
                    Empleados emp = new Empleados();
                    emp.Idempleado = txtNumEmpleado.Text;
                    emp.Nombrecompleto = txtNombre.Text;
                    emp.RFC = txtRFC.Text;
                    emp.CURP = txtCURP.Text;
                    emp.Edad =  Convert.ToInt32(nudEdad.Value);
                    emp.Password = txtContrasena.Text;
                    emp.Correo = txtCorreo.Text;
                    emp.Direccion = txtDireccion.Text;
                    emp.Rol = cbxRol.SelectedItem.ToString();
                    emp.Telefono = txtTelefono.Text;
                    emp.Estatus = true;
                    emp.fechaContratacion = dtpContrato.Text;
                    if (Verificar())
```

```
                    {
                        Guardado = new DAOEmpleados().AgregarEmpleado(emp);
                        if (Guardado == 0)
                        {
                            MessageBox.Show("Usuario agregado correctamente", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                            this.Close();
                        }
                        else
                        {
                            MessageBox.Show("A surgido un problema, inténtelo de nuevo más tardé", "", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                        }
                    }
                }
                else if (OP == 2)
                {
                    Empleados emp = new Empleados();
                    emp.Idempleado = txtNumEmpleado.Text;
                    emp.Nombrecompleto = txtNombre.Text;
                    emp.RFC = txtRFC.Text;
                    emp.CURP = txtCURP.Text;
                    emp.Edad = Convert.ToInt32(nudEdad.Value);
                    emp.Password = txtContrasena.Text;
                    emp.Correo = txtCorreo.Text;
                    emp.Direccion = txtDireccion.Text;
                    emp.Rol = cbxRol.SelectedItem.ToString();
                    emp.Telefono = txtTelefono.Text;
                    if (Verificar())
                    {
                        Modificado = new DAOEmpleados().ModificarEmpleado(emp);
                        if (Modificado)
                        {
                            MessageBox.Show("Usuario modificado correctamente", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                            this.Close();
                        }
                        else
                        {
                            MessageBox.Show("A surgido un problema, inténtelo de nuevo más tardé", "", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                        }
                    }
                }

        }

        /// <summary>
        /// Evento de boton cancelar, al dar click al boton cierra la ventana y regresa al menu principal
        /// de Empleados.
        /// </summary>
        private void btnCancelar_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Formulario de Añadir y Modificar Instalaciones

```
using Datos;
using Modelos;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Reflection.Emit;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SGMDT_Wifimex
{
    /// <summary>
    /// Formulario que agrega y modifica los registros de la tabla Instalaciones
    /// </summary>
    public partial class frmAgregarModificarInsta : Form
    {
        /// <summary>
        /// Variables que sirven para validar la opcion que se mostrara en el formulario
        /// ya se Agregar o modificar.
        /// </summary>
        private int OP;
        private string ID;
        public int Guardado { get; set; }
        public bool Modificado { get; set; }

        /// <summary>
        /// Metodo que sirve para validar que las cajas de texto del formulario tengan
        /// el formato correcto que se solicita
        /// </summary>
        /// <returns>
        /// regresa un true si los formatos de las cajas de texto son validos en caso de
        /// lo contrario regresa un false.
        /// </returns>
        public bool Verificar()
        {
            if (!Regex.IsMatch(txtInstalacion.Text, "^[A-Za-z]{4}[0-9]{6}$"))
            {
                errorProvider1.SetError(txtInstalacion, "El formato debe contener 4 letras y 6 numeros en este orden");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }
            if (!Regex.IsMatch(txtEmpleado.Text, "^[A-Za-z]{4}[0-9]{6}$"))
            {
                errorProvider1.SetError(txtEmpleado, "El formato debe contener 4 letras y 6 numeros en este orden");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }
            if (!Regex.IsMatch(txtContrato.Text, "^[A-Za-z]{4}[0-9]{6}$"))
            {
                errorProvider1.SetError(txtContrato, "El formato debe contener 4 letras y 6 numeros en este orden");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }


            return true;
        }

        /// <summary>
        /// Costrcutor del formulario donde cargan todos los controles que
        /// estan dentro del formulario.
        /// </summary>
        /// <param name="id">
        /// Parametro que recibe el constructor en caso de que se dé la opcion de modificar
        /// un registro para buscar el registro a modificar
        /// </param>
        /// <param name="op">
        /// Parametro que indica cuales controles tiene que cargar el formulario
        /// este indica si el formulario tiene que ponerse en modo de agregar un
        /// nuevo registro o en modo de modificar un registro ya existente.
        /// </param>
        public frmAgregarModificarInsta(int op, string id)
        {
            InitializeComponent();
            ID = id;
            OP = op;
```

```
            if (op == 1)
            {
                this.Text = "Agregar";
            }
            // se cargan los datos a actualizar
            else if (op == 2)
            {
                this.Text = "Modificar";
                Instalaciones ints = new DAOInstalaciones().ObtenerUnaInstalacion(id);
                txtInstalacion.Text = ints.IdInstalacion;
                txtEmpleado.Text = ints.idEmpelado;
                txtContrato.Text = ints.idContratos;
                dtpInstalacion.Text = ints.fechaInstalacin;
                dtpInstalacion.Enabled = false;
                txtInstalacion.Enabled = false;
            }
        }

        /// <summary>
        /// Evento click de boton, al dar click mandara y guardara el nuevo registro o modificara
        /// un registro ya existente.
        /// En caso de que la opcion seleccionada sea agregar se hace una referencia al modelo de Instalaciones
        /// así como al DAO Instalaciones para poder utilizar el metodo de agregar un nuevo registro.
        /// Si la opcion seleccionada es modificar también hará referencias a las mismas clases ya
        /// mencionadas, con el cambio de que ahora llamara al metodo de actualizar un registro.
        /// Al finalizar lanzar un mensaje de registro exitoso o fallo de registro, al cerrar
        /// el mensaje regresara al menu principal de Instalaciones.
        /// </summary>
        private void btnGuardar_Click(object sender, EventArgs e)
        {

                if (OP == 1)
                {
                    Instalaciones Inst = new Instalaciones();
                    Inst.IdInstalacion = txtInstalacion.Text;
                    Inst.idEmpelado = txtEmpleado.Text;
                    Inst.idContratos = txtContrato.Text;
                    Inst.Estatus = true;
                    Inst.fechaInstalacin = dtpInstalacion.Text;
                if (Verificar())
                {
                    Guardado = new DAOInstalaciones().AgregarInstlacion(Inst);
                    if (Guardado == 0)
                    {
                        MessageBox.Show("Instalacion agregada correctamente", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        this.Close();
                    }
                    else
                    {
                        MessageBox.Show("A surgido un problema, inténtelo de nuevo más tardé", "", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                    }
                }
                }
                else if (OP == 2)
                {
                    Instalaciones Inst = new Instalaciones();
                    Inst.IdInstalacion = txtInstalacion.Text;
                    Inst.idEmpelado = txtEmpleado.Text;
                    Inst.idContratos = txtContrato.Text;
                    Inst.fechaInstalacin = dtpInstalacion.Text;
                if (Verificar())
                {
                    Modificado = new DAOInstalaciones().ModificarInstalacion(Inst);
                    if (Modificado)
                    {
                        MessageBox.Show("Instalacion modificada correctamente", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        this.Close();
                    }
                    else
                    {
                        MessageBox.Show("A surgido un problema, inténtelo de nuevo más tardé", "", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                    }
                }
                }

        }

        /// <summary>
        /// Evento de boton cancelar, al dar click al boton cierra la ventana y regresa al menu principal
        /// de Instalaciones.
        /// </summary>
        private void btnCancelar_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Formulario menu principal Instalaciones

```
using Datos;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SGMDT_Wifimex
{
    /// <summary>
    /// Formulario de menu principal de Empleados, donde se darán las opciones de agregar, actualizar y
    /// Buscar registros de la tabla de Instalaciones.
    /// </summary>
    public partial class frmInstalaciones : Form
    {
        /// <summary>
        /// Constructor del formulario donde se cargarán los controles que estarán visibles en
        /// el formulario, también cargara los datos de la tabla de Instalaciones en una tabla para su
        /// visualización
        /// </summary>
        public frmInstalaciones()
        {
            InitializeComponent();
            CargarTablaAc();
        }

        /// <summary>
        /// Mentodo que sirve para cargar los datos de la tabla Instalaciones en una tabla del sistema
        /// haciendo una instancia del DAO de Instalaciones al metodo obtener todos los registros
        /// que estén en estado activos.
        /// </summary>
        public void CargarTablaAc()
        {
            dgvInstalaciones.DataSource = new DAOInstalaciones().ObtenerInstalacionesActivos();
            dgvInstalaciones.Columns["IdInstalacion"].HeaderText = "ID Instalaciones";
            dgvInstalaciones.Columns["fechaInstalacin"].HeaderText = "Fecha de Instalacion";
            dgvInstalaciones.Columns["idEmpelado"].HeaderText = "ID Empleado";
            dgvInstalaciones.Columns["idContratos"].HeaderText = "ID Conntrato";
            dgvInstalaciones.Columns["Estatus"].Visible = false;
        }

        /// <summary>
        /// Mentodo que sirve para cargar los datos de la tabla Instalaciones en una tabla del sistema
        /// haciendo una instancia del DAO de Instalaciones al metodo obtener todos los registros
        /// en estado inactivos.
        /// </summary>
        public void CargarTablaIn()
        {
            dgvInstalaciones.DataSource = new DAOInstalaciones().ObtenerInstalacionesInactivos();
            dgvInstalaciones.Columns["IdInstalacion"].HeaderText = "ID Instalaciones";
            dgvInstalaciones.Columns["fechaInstalacin"].HeaderText = "Fecha de Instalacion";
            dgvInstalaciones.Columns["idEmpelado"].HeaderText = "ID Empleado";
            dgvInstalaciones.Columns["idContratos"].HeaderText = "ID Conntrato";
            dgvInstalaciones.Columns["Estatus"].Visible = false;
        }

        /// <summary>
        /// Mentodo que sirve para cargar los datos de la tabla Instalaciones en una tabla del sistema
        /// haciendo una instancia del DAO de Instalaciones al metodo obtener todos los registros.
        /// </summary>
        public void CargarTabla()
        {
            dgvInstalaciones.DataSource = new DAOInstalaciones().ObtenerInstalaciones();
            dgvInstalaciones.Columns["IdInstalacion"].HeaderText = "ID Instalaciones";
            dgvInstalaciones.Columns["fechaInstalacin"].HeaderText = "Fecha de Instalacion";
            dgvInstalaciones.Columns["idEmpelado"].HeaderText = "ID Empleado";
            dgvInstalaciones.Columns["idContratos"].HeaderText = "ID Conntrato";
            dgvInstalaciones.Columns["Estatus"].Visible = false;
        }

        /// <summary>
        /// Mentodo que sirve para cargar los datos de la tabla Instalaciones en una tabla del sistema
        /// haciendo una instancia del DAO de Instalaciones al metodo obtener todos los registros.
        /// </summary>
        public void CargarTablaIns()
        {
            dgvInstalaciones.DataSource = new DAOInstalaciones().ObtenerInstalacion(txtBuscar.Text);
            dgvInstalaciones.Columns["IdInstalacion"].HeaderText = "ID Instalaciones";
            dgvInstalaciones.Columns["fechaInstalacin"].HeaderText = "Fecha de Instalacion";
            dgvInstalaciones.Columns["idEmpelado"].HeaderText = "ID Empleado";
            dgvInstalaciones.Columns["idContratos"].HeaderText = "ID Conntrato";
            dgvInstalaciones.Columns["Estatus"].Visible = false;
        }

        /// <summary>
        /// evento del boton Añadir, al dar click en este boton abrira la ventana de
```

```
        /// agrega/modificar un registro de Instalaciones. haciendo una instancia del
        /// formulario de agregar/modificar Instalaciones, cundo se finalice
        /// el trámite de añadir un nuevo registro se volverá a cargar la tabla
        /// con los nuevos registros
        /// </summary>
        private void btnAgregar_Click(object sender, EventArgs e)
        {
            frmAgregarModificarInsta frm = new frmAgregarModificarInsta(1, "0");
            frm.ShowDialog();
            if (frm.Guardado == 0)
            {
                CargarTabla();
            }
        }

        /// <summary>
        /// evento del boton Modificar, al seleccionar un registro
        /// de la tabla y dar click en este boton abrira la ventana de
        /// agrega/modificar un registro de Instalaciones. haciendo una instancia del
        /// formulario de agregar/modificar Instalaciones, cundo se finalice
        /// el trámite de modificar un registro se volverá a cargar la tabla
        /// con los registros actualizados.
        /// </summary>
        private void btnActualizar_Click(object sender, EventArgs e)
        {
            if (dgvInstalaciones.SelectedRows.Count > 0)
            {
                string idCategoria = dgvInstalaciones.SelectedRows[0].Cells["IdInstalacion"].Value.ToString();
                frmAgregarModificarInsta frm = new frmAgregarModificarInsta(2, idCategoria);
                frm.ShowDialog();
                if (frm.Modificado)
                {
                    CargarTabla();
                }
            }
        }

        /// <summary>
        /// Evento de boton buscar, cuando se da click a este boton
        /// mostrara todos los registros que coincida con lo que está en la
        /// caja de texto de la tabla empleados sin importar
        /// si están activos o inactivos
        /// </summary>
        private void btnBuscar_Click(object sender, EventArgs e)
        {
            CargarTablaIns();
        }
    }
}
```

Formulario menu principal de Productos

```
using Datos;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SGMDT_Wifimex
{
    // <summary>
    /// Formulario de menu principal de Productos, donde se darán las opciones de agregar, actualizar,
    /// Buscar y eliminar registros de la tabla de Productos.
    /// </summary>
    public partial class frmProductos : Form
    {
        /// <summary>
        /// Constructor del formulario donde se cargarán los controles que estarán visibles en
        /// el formulario, también cargara los datos de la tabla de Productos en una tabla para su
        /// visualización
        /// </summary>
        public frmProductos()
        {
            InitializeComponent();
            CargarTablaAc();
        }

        /// <summary>
        /// evento del boton Añadir, al dar click en este boton abrira la ventana de
        /// agrega/modificar un registro de Productos. haciendo una instancia del
        /// formulario de agregar/modificar Productos, cundo se finalice
        /// el trámite de añadir un nuevo registro se volverá a cargar la tabla
        /// con los nuevos registros
        /// </summary>
        private void btnAgregar_Click(object sender, EventArgs e)
        {
            frmAgregar_ModificarProductos frm = new frmAgregar_ModificarProductos(1, "0");
            frm.ShowDialog();
            if (frm.Guardado > 0)
            {
                CargarTablaAc();
            }
        }

        /// <summary>
        /// evento del boton Modificar, al seleccionar un registro
        /// de la tabla y dar click en este boton abrira la ventana de
        /// agrega/modificar un registro de Productos. haciendo una instancia del
        /// formulario de agregar/modificar Productos, cundo se finalice
        /// el trámite de modificar un registro se volverá a cargar la tabla
        /// con los registros actualizados.
        /// </summary>
        private void btnActualizar_Click(object sender, EventArgs e)
        {
            if (dgvProductos.SelectedRows.Count > 0)
            {
                string codigoBarra = dgvProductos.SelectedRows[0].Cells["codigoBarra"].Value.ToString();
                frmAgregar_ModificarProductos frm = new frmAgregar_ModificarProductos(2, codigoBarra);
                frm.ShowDialog();
                if (frm.Modificado)
                {
                    CargarTablaAc();
                }
            }
        }

        /// <summary>
        /// Evento del boton eliminar, al dar click a eliminar se eliminará el registro seleccionado
        /// de la tabla. Al seleccionar un registro de la tabla y dar click a eliminar emergerá un mensaje
        /// donde se pregunta si se quiere eliminar el registro, al seleccionar si el registro se eliminara
        /// de la base de datos y cargara la tabla con los registros actualizados, en caso de seleccionar
        /// no la acción se abortara y no habrá ningún cambio.
        /// </summary>
        private void btnEliminar_Click(object sender, EventArgs e)
        {
            if (dgvProductos.SelectedRows.Count > 0)
            {
                string codigo = dgvProductos.SelectedRows[0].Cells["codigoBarra"].Value.ToString();
                DialogResult resp = MessageBox.Show("Estas a punto de borrar a " +
                                    (new DAOProductos().ObtenerUnProducto(codigo).nomProducto) +
                                    " del registro, ¿Deseas continuar?", "",
                                    MessageBoxButtons.YesNo, MessageBoxIcon.Question);
                if (resp == DialogResult.Yes)
                {
                    if (new DAOProductos().EliminarProducto(codigo))
                    {
```

```
                        MessageBox.Show("Se ha borrado exitosamente el producto", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        CargarTablaAc();
                    }
                }
            }
            else
            {
                MessageBox.Show("Selecciona un registro primero", "", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }

        /// <summary>
        /// Evento de boton buscar, cuando se da click a este boton
        /// mostrara todo el registro que coincida con lo que está en la
        /// caja de texto de la tabla Productos sin importar
        /// si están activos o inactivos
        /// </summary>
        private void btnBuscar_Click(object sender, EventArgs e)
        {
            DAOProductos prov = new DAOProductos();
            dgvProductos.DataSource = new DAOProductos().Buscar(txtBuscar.Text);
            dgvProductos.Columns["codigoBarra"].Visible = false;
            dgvProductos.Columns["nomProducto"].HeaderText = "Producto";
            dgvProductos.Columns["fechaRegistro"].HeaderText = "Registro";
            dgvProductos.Columns["idProveedores"].Visible = false;
            dgvProductos.Columns["nomProveedor"].HeaderText = "Proveedor";
            dgvProductos.Columns["Estatus"].Visible = false;
            txtBuscar.Text = "";
            MessageBox.Show("Búsqueda exitosa", "", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        /// <summary>
        /// Mentodo que sirve para cargar los datos de la tabla Productos en una tabla del sistema
        /// haciendo una instancia del DAO de Productos al metodo obtener todos los registros
        /// que estén en estado activos.
        /// </summary>
        public void CargarTablaAc()
        {
            dgvProductos.DataSource = new DAOProductos().ObtenerProductosActivos();
            dgvProductos.Columns["codigoBarra"].Visible = false;
            dgvProductos.Columns["nomProducto"].HeaderText = "Producto";
            dgvProductos.Columns["fechaRegistro"].HeaderText = "Registro";
            dgvProductos.Columns["idProveedores"].Visible = false;
            dgvProductos.Columns["nomProveedor"].HeaderText = "Proveedor";
            dgvProductos.Columns["Estatus"].Visible = false;
        }
    }
}
```

Formulario de Añadir y Modificar Productos

```
using Datos;
using Modelos;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SGMDT_Wifimex
{
    /// <summary>
    /// Formulario que agrega y modifica los registros de la tabla Productos
    /// </summary>
    public partial class frmAgregar_ModificarProductos : Form
    {
        /// <summary>
        /// Variables que sirven para validar la opcion que se mostrara en el formulario
        /// ya se Agregar o modificar.
        /// </summary>
        private int OP;
        private string codigo;
        public int Guardado { get; set; }
        public bool Modificado { get; set; }

        /// <summary>
        /// Metodo que sirve para validar que las cajas de texto del formulario tengan
        /// el formato correcto que se solicita
        /// </summary>
        /// <returns>
        /// regresa un true si los formatos de las cajas de texto son válidos en caso de
        /// lo contrario regresa un false.
        /// </returns>
        public bool Verificar()
        {
            if (!Regex.IsMatch(txtCodigo.Text, "^[0-9]{12}$"))
            {
                errorProvider1.SetError(txtCodigo, "El formato debe contener 12 numeros");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }
            if (!Regex.IsMatch(txtNombre.Text, @"^[A-Za-z][A-Za-z .0-9]{1,}$"))
            {
                errorProvider1.SetError(txtNombre, "El nombre no puede contener números o caracteres especiales");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }


            return true;
        }

        /// <summary>
        /// Constructor del formulario donde cargan todos los controles que
        /// están dentro del formulario.
        /// </summary>
        /// <param name="codigoBarra">
        /// Parametro que recibe el constructor en caso de que se de la opcion de modificar
        /// un registro para buscar el registro a modificar
        /// </param>
        /// <param name="op">
        /// Parametro que indica cuales controles tiene que cargar el formulario
        /// este indica si el formulario tiene que ponerse en modo de agregar un
        /// nuevo registro o en modo de modificar un registro ya existente.
        /// </param>
        public frmAgregar_ModificarProductos(int op, string codigoBarra)
        {
            InitializeComponent();
            this.OP = op;
            this.codigo = codigoBarra;
            if (op == 1)
            {
                this.Text = "Registrar";
                lblTitulo.Text = "Registrar productos";
                List<Proveedores> prov = new DAOProveedores().ObtenerProveedoresActivos();
                cbxProveedor.DataSource = prov;
                cbxProveedor.DisplayMember = "nomProveedores";
                cbxProveedor.ValueMember = "idProveedores";
                //cbxProveedor.SelectedIndex = 1;
            }
```

```
            else if (op == 2)
            {
                this.Text = "Modificar";
                Productos ints = new DAOProductos().ObtenerUnProducto(codigo);
                lblTitulo.Text = "Modificar productos";
                txtCodigo.Text = ints.codigoBarra.ToString();
                txtCodigo.Enabled = false;
                txtNombre.Text = ints.nomProducto;
                dtpFecha.Enabled = false;
                List<Proveedores> prov = new DAOProveedores().ObtenerProveedoresActivos();
                cbxProveedor.DataSource = prov;
                cbxProveedor.DisplayMember = "nomProveedores";
                cbxProveedor.ValueMember = "idProveedores";
                cbxProveedor.SelectedValue=ints.idProveedores;
            }
        }

        /// <summary>
        /// Evento click de boton, al dar click mandara y guardara el nuevo registro o modificara
        /// un registro ya existente.
        /// En caso de que la opcion seleccionada sea agregar se hace una referencia al modelo de Productos
        /// así como al DAO Productos para poder utiliazar el metodo de agregar un nuevo registro.
        /// Si la opcion seleccionada es modificar también hará referencias a las mismas clases ya
        /// mencionadas, con el cambio de que ahora llamara al metodo de actualizar un registro.
        /// Al finalizar lanzar un mensaje de registro exitoso o fallo de registro, al cerrar
        /// el mensaje regresara al menu principal de Productos.
        /// </summary>
        private void btnGuardar_Click(object sender, EventArgs e)
        {
            if (txtCodigo.Text.Length > 0 & txtNombre.Text.Length > 0)
            {
                if (OP == 1)
                {
                    Productos Inst = new Productos();
                    Inst.codigoBarra = txtCodigo.Text;
                    Inst.nomProducto = txtNombre.Text;
                    Inst.fechaRegistro = dtpFecha.Text;
                    Inst.idProveedores = cbxProveedor.SelectedValue.ToString();
                    Inst.nomProveedor = cbxProveedor.SelectedText;
                    Inst.Estatus = true;
                    if (Verificar())
                    {
                        Guardado = new DAOProductos().AgregarProducto(Inst);
                        if (Guardado > 0)
                        {
                            MessageBox.Show("Se ha guardado exitosamente el producto", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                            this.Close();
                        }
                        else
                        {
                            MessageBox.Show("Producto ya registrado", "", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                        }
                    }
                    else
                    {
                        MessageBox.Show("Rellene todos los campos", "", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }
                else if (OP == 2)
                {
                    Productos Inst = new Productos();
                    Inst.codigoBarra = txtCodigo.Text;
                    Inst.nomProducto = txtNombre.Text;
                    Inst.fechaRegistro = dtpFecha.Text;
                    Inst.idProveedores = cbxProveedor.SelectedValue.ToString();
                    Inst.nomProveedor = cbxProveedor.SelectedText;
                    Inst.Estatus = true;
                    if (Verificar())
                    {
                        Modificado = new DAOProductos().ModificarProducto(Inst);
                        if (Modificado)
                        {
                            MessageBox.Show("Se ha actualizado exitosamente el producto", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                            this.Close();
                        }
                        else
                        {
                            MessageBox.Show("A surgido un problema, inténtelo de nuevo más tardé", "", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                        }
                    }
                    else
                    {
                        MessageBox.Show("Rellene todos los campos", "", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }
            }
        }

        /// <summary>
        /// Evento de boton cancelar, al dar click al boton cierra la ventana y regresa al menu principal
```

```
        /// de Productos.
        /// </summary>
        private void btnCancelar_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Formulario menu principal de Proveedores

```
using Datos;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SGMDT_Wifimex
{
    // <summary>
    /// Formulario de menu principal de Proveedores, donde se darán las opciones de agregar, actualizar,
    /// Buscar y eliminar registros de la tabla de Proveedores.
    /// </summary>
    public partial class frmProveedores : Form
    {
        /// <summary>
        /// Constructor del formulario donde se cargarán los controles que estarán visibles en
        /// el formulario, también cargara los datos de la tabla de Proveedores en una tabla para su
        /// visualización
        /// </summary>
        public frmProveedores()
        {
            InitializeComponent();
            CargarTablaAc();
        }

        /// <summary>
        /// Mentodo que sirve para cargar los datos de la tabla Proveedores en una tabla del sistema
        /// haciendo una instancia del DAO de Proveedores al metodo obtener todos los registros
        /// que esten en estado activos.
        /// </summary>
        public void CargarTablaAc()
        {
            dgvProveedores.DataSource = new DAOProveedores().ObtenerProveedoresActivos();
            dgvProveedores.Columns["idProveedores"].Visible = false;
            dgvProveedores.Columns["nomProveedores"].HeaderText = "Proveedor";
            dgvProveedores.Columns["fechaRegistro"].HeaderText = "Registro";
            dgvProveedores.Columns["Estatus"].Visible = false;
        }

        /// <summary>
        /// evento del boton Añadir, al dar click en este boton abrira la ventana de
        /// agrega/modificar un registro de Proveedores. haciendo una instancia del
        /// formulario de agregar/modificar Proveedores, cundo se finalice
        /// el trámite de añadir un nuevo registro se volverá a cargar la tabla
        /// con los nuevos registros
        /// </summary>
        private void btnAgregar_Click(object sender, EventArgs e)
        {
            frmAgregar_ModificarProveedores frm = new frmAgregar_ModificarProveedores(1, "0");
            frm.ShowDialog();
            if (frm.Guardado > 0)
            {
                CargarTablaAc();
            }
        }

        /// <summary>
        /// evento del boton Modificar, al seleccionar un registro
        /// de la tabla y dar click en este boton abrira la ventana de
        /// agrega/modificar un registro de Proveedores. haciendo una instancia del
        /// formulario de agregar/modificar Proveedores, cundo se finalice
        /// el trámite de modificar un registro se volverá a cargar la tabla
        /// con los registros actualizados.
        /// </summary>
        private void btnModificar_Click(object sender, EventArgs e)
        {
            if (dgvProveedores.SelectedRows.Count > 0)
            {
                string idProveedor = dgvProveedores.SelectedRows[0].Cells["idProveedores"].Value.ToString();
                frmAgregar_ModificarProveedores frm = new frmAgregar_ModificarProveedores(2, idProveedor);
                frm.ShowDialog();
                if (frm.Modificado)
                {
                    CargarTablaAc();
                }
            }
        }

        /// <summary>
        /// Evento del boton eliminar, al dar click a eliminar se eliminará el registro seleccionado
        /// de la tabla. Al seleccionar un registro de la tabla y dar click a eliminar emergerá un mensaje
        /// donde se pregunta si se quiere eliminar el registro, al seleccionar si el registro se eliminara
        /// de la base de datos y cargara la tabla con los registros actualizados, en caso de seleccionar
        /// no la acción se abortará y no habrá ningún cambio.
        /// </summary>
```

```
        private void btnEliminar_Click(object sender, EventArgs e)
        {
            if (dgvProveedores.SelectedRows.Count > 0)
            {
                String idProveedor = dgvProveedores.SelectedRows[0].Cells["idProveedores"].Value.ToString();
                DialogResult resp = MessageBox.Show("Estas a punto de borrar a " +
                                    (new DAOProveedores().ObtenerUnProveedor(idProveedor).nomProveedores) +
                                    " del registro, ¿Deseas continuar?", "",
                                    MessageBoxButtons.YesNo, MessageBoxIcon.Question);
                if (resp == DialogResult.Yes)
                {
                    if (new DAOProveedores().EliminarProveedor(idProveedor))
                    {
                        MessageBox.Show("Se ha borrado exitosamente el proveedor", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        CargarTablaAc();
                    }
                }
            }
            else
            {
                MessageBox.Show("Selecciona un registro primero", "",MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }

        /// <summary>
        /// Evento de boton buscar, cuando se da click a este boton
        /// mostrara todos los registros que coincida con lo que está en la
        /// caja de texto de la tabla Proveedores sin importar
        /// si están activos o inactivos
        /// </summary>
        private void btnBuscar_Click(object sender, EventArgs e)
        {
            DAOProveedores prov = new DAOProveedores();
            dgvProveedores.DataSource = new DAOProveedores().Buscar(txtBuscar.Text);
            dgvProveedores.Columns["idProveedores"].Visible = false;
            dgvProveedores.Columns["nomProveedores"].HeaderText = "Proveedor";
            dgvProveedores.Columns["fechaRegistro"].HeaderText = "Registro";
            dgvProveedores.Columns["Estatus"].Visible = false;
            txtBuscar.Text = "";
            MessageBox.Show("Búsqueda exitosa", "", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
}
```

Formulario de Añadir y Modificar Proveedores

```
using Datos;
using Modelos;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SGMDT_Wifimex
{
    /// <summary>
    /// Formulario que agrega y modifica los registros de la tabla Proveedores
    /// </summary>
    public partial class frmAgregar_ModificarProveedores : Form
    {
        /// Variables que sirven para validar la opcion que se mostrara en el formulario
        /// ya se Agregar o modificar.
        /// </summary>
        private int OP;
        private string ID;
        public int Guardado { get; set; }
        public bool Modificado { get; set; }

        /// <summary>
        /// Metodo que sirve para validar que las cajas de texto del formulario tengan
        /// el formato correcto que se solicita
        /// </summary>
        /// <returns>
        /// regresa un true si los formatos de las cajas de texto son válidos en caso de
        /// lo contrario regresa un false.
        /// </returns>
        public bool Verificar()
        {
            if (!Regex.IsMatch(txtClave.Text, "^[A-Za-z]{4}[0-9]{6}$"))
            {
                errorProvider1.SetError(txtClave, "El formato debe contener 4 letras y 6 numeros en este orden");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }
            if (!Regex.IsMatch(txtNombre.Text, @"^[A-Za-z][A-Za-z .]{1,}[A-Za-z]$"))
            {
                errorProvider1.SetError(txtNombre, "El nombre no puede contener números o caracteres especiales");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }
            if (!Regex.IsMatch(txtRFC.Text, @"^[A-Z]{4}\d{6}([A-Z0-9]{2}[A-Z0-9]{1})?$"))
            {
                errorProvider1.SetError(txtRFC, "Verifique que su RFC este correcto");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }
            if (!Regex.IsMatch(txtDireccion.Text, @"^[A-Za-z][A-Za-z .]{1,}\s+#\d+(-\d+)?$"))
            {
                errorProvider1.SetError(txtDireccion, "Siga al formato adecuado , ejemplo : 'Calle #123', en caso de ser
departamento agregue '-Número interior'");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }
            if (!Regex.IsMatch(txtTelefono.Text, @"^[0-9]{10}?$"))
            {
                errorProvider1.SetError(txtTelefono, "Verifique el número contenga 10 dígitos");
                return false;
            }
            else
            {
                errorProvider1.Clear();
            }
            if (!Regex.IsMatch(txtCorreo.Text, @"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"))
            {
                errorProvider1.SetError(txtCorreo, "Verifique en correo este correctamente escrito y sin espacios");
                return false;
            }
            else
```

```
            {
                errorProvider1.Clear();
            }


            return true;
        }

        /// <summary>
        /// Constructor del formulario donde cargan todos los controles que
        /// están dentro del formulario.
        /// </summary>
        /// <param name="id">
        /// Parametro que resive el constructor en caso de que se dé la opcion de modificar
        /// un registro para buscar el registro a modificar
        /// </param>
        /// <param name="op">
        /// Parametro que indica cuales controles tiene que cargar el formulario
        /// este indica si el formulario tiene que ponerse en modo de agregar un
        /// nuevo registro o en modo de modificar un registro ya existente.
        /// </param>
        public frmAgregar_ModificarProveedores(int op, string id)
        {
            InitializeComponent();
            this.OP = op;
            if (op == 1)
            {
                this.Text = "Registrar";
                lblTitulo.Text = "Registrar proveedores";
            }
            else if (op == 2)
            {
                this.Text = "Modificar";
                lblTitulo.Text = "Modificar proveedores";
                Proveedores ints = new DAOProveedores().ObtenerUnProveedor(id);
                txtClave.Text = ints.idProveedores;
                txtClave.Enabled = false;
                txtNombre.Text = ints.nomProveedores;
                txtRFC.Text = ints.RFC;
                txtDireccion.Text = ints.Direccion;
                txtTelefono.Text = ints.Telefono;
                txtCorreo.Text = ints.Correo;
                dtpFecha.Enabled = false;
            }
        }

        /// <summary>
        /// Evento click de boton, al dar click mandara y guardara el nuevo registro o modificara
        /// un registro ya existente.
        /// En caso de que la opcion seleccionada sea agregar se hace una referencia al modelo de Proveedores
        /// así como al DAO Proveedores para poder utilizar el metodo de agregar un nuevo registro.
        /// Si la opcion seleccionada es modificar también hará referencias a las mismas clases ya
        /// mencionadas, con el cambio de que ahora llamara al metodo de actualizar un registro.
        /// Al finalizar lanzar un mensaje de registro exitoso o fallo de registro, al cerrar
        /// el mensaje regresara al menu principal de Proveedores.
        /// </summary>
        private void btnGuardar_Click(object sender, EventArgs e)
        {

                if (OP == 1)
                {
                    Proveedores Inst = new Proveedores();
                    Inst.idProveedores = txtClave.Text;
                    Inst.nomProveedores = txtNombre.Text;
                    Inst.RFC = txtRFC.Text;
                    Inst.Direccion = txtDireccion.Text;
                    Inst.Telefono = txtTelefono.Text;
                    Inst.Correo = txtCorreo.Text;
                    Inst.fechaRegistro = dtpFecha.Text;
                    Inst.Estatus = true;
                    if (Verificar())
                    {
                    Guardado = new DAOProveedores().AgregarProveedor(Inst);
                    if (Guardado > 0)
                    {
                        MessageBox.Show("Se ha guardado exitosamente el proveedor", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        this.Close();
                    }
                    else
                    {
                        MessageBox.Show("A surgido un problema, inténtelo de nuevo más tardé", "", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                    }
                }
                    else
                {
                    MessageBox.Show("Rellene todos los campos correctamente","", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                }
                }
                else if (OP == 2)
                {
```

```
                    Proveedores Inst = new Proveedores();
                    Inst.idProveedores = txtClave.Text;
                    Inst.nomProveedores = txtNombre.Text;
                    Inst.RFC = txtRFC.Text;
                    Inst.Direccion = txtDireccion.Text;
                    Inst.Telefono = txtTelefono.Text;
                    Inst.Correo = txtCorreo.Text;
                    Inst.fechaRegistro = dtpFecha.Text;
                    Inst.Estatus = true;
                if (Verificar())
                {
                    Modificado = new DAOProveedores().ModificarProveedor(Inst);
                    if (Modificado)
                    {
                        MessageBox.Show("Se ha actualizado exitosamente el proveedor.", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                        this.Close();
                    }
                    else
                    {
                        MessageBox.Show("A surgido un problema, inténtelo de nuevo más tardé", "", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                    }
                }
                else
                {
                    MessageBox.Show("Rellene todos los campos correctamente", "",MessageBoxButtons.OK,
MessageBoxIcon.Information);
                }
                }

        }

        /// <summary>
        /// Evento de boton cancelar, al dar click al boton cierra la ventana y regresa al menu principal
        /// de Proveedores.
        /// </summary>
        private void btnCancelar_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```